MASTER EN SCIENCES INFORMATIQUES

Interprétation Abstraite Tabulée de Programmes Logiques et Analyse de Groundness

Meurisse, Renaud

Award date:
1993

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grandgagnage, 21b
B-5000 NAMUR

Interprétation Abstraite Tabulée de Programmes Logiques et Analyse de *Groundness*

Renaud MEURISSE

Promoteur : Baudouin Le Charlier

Mémoire présenté en vue
de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1992 - 1993

Résumé

L'interprétation abstraite est une technique avancée d'analyse statique destinée, entre autre, au perfectionnement de compilateurs. Elle est particulièrement utilisée dans le cadre de programmes logiques, où l'efficacité fait défaut.

Dans ce travail, nous rappellerons ce qu'est l'interprétation abstraite et quel est son rôle en programmation logique.

Les méthodes ainsi décrites serviront à l'introduction d'un algorithme d'interprétation abstraite tabulée, basé sur une sémantique opérationnelle. Cet algorithme sera tout d'abord présenté de manière théorique, et fera ensuite l'objet d'une application pratique.

Le domaine abstrait PROP fera aussi l'objet d'un chapitre et sera implémenté pour être utilisé avec le programme d'interprétation. Seront également exposées certaines opérations de gestion de fonctions propositionnelles, indispensables à la mise au point du domaine, ces opérations pouvant néanmoins être utilisées indépendamment de toute interprétation abstraite ou domaine abstrait.

Enfin, le programme élaboré fonctionnera sur des exemples connus de programmes Prolog, et une série de résultats seront présentés.

Abstract

Abstract interpretation is an advanced technique for static analysis used, mainly, for compilator's improvement. It is used especially with logic programs, where inefficacy is the rule.

This work will provide a reminding of abstract interpretation and the relations with logic programming.

The method described will contribute to carry on a tabled abstract interpretion algorithm based on operational semantic. This algorithm will be first shown theorically, and then will be the object of an application.

The abstract domain PROP will also be described and implemented in such a way to be used in the interpretation program. We will present some procedures that can handle propositional functions, that we need for the development of this domain. This procedures may be used separetly of any abstract interpretation or abstract domain.

Then the program will be tested up on well-known Prolog programs. A serie of results will be presented.

Remerciements

Sur la page de garde, on trouve un nom: le mien.

Pourtant je ne suis pas seul dans la "paternité" de ce travail.

Je tiens donc à remercier le professeur Baudouin Le Charlier, mon promoteur , pour l'indication des voies à suivre, et ses explications fructueuses.

Je remercie également le professeur Gilberto Filé, sans qui mon programme n'existerait pas, ainsi que tous les assistants, doctorants et étudiants du département de mathématique de l'université de Padoue: Giuseppe, Sabina, Luca, Maria,...

Je tiens encore à remercier toutes les personnes qui, de près ou de loin, ont pu m'aider dans mes travaux. Il m'est impossible de tous les nommer, mais j'aimerais qu'ils sachent que je leur dois beaucoup.

Mais je tiens tout particulièrement à remercier mon épouse pour sa longue compréhension et pour tous les conseils précieux qu'elle m'a apportées.

Et puis, il y a Aline.
Si son discours ne fut jamais très explicite, sa présence était importante et m'a encouragé sur le chemin de l'accomplissement.

Renaud

Table des Matières

I. Introduction

A la base de ce travail figure Prolog.
Prolog est un langage de programmation logique, qui a été l'objet de nombreuses recherches et études.
Son succès certain provient principalement de ses caractéristiques déclaratives.

Parmi les nombreux livres et documents analysant les avantages, inconvénients et utilisations de cette nouvelle méthodologie de programmation nous pouvons retenir deux ouvrages: "Foundation of the Logic Programming" de J.W.Lloyd, et "The Art of Prolog" de L.Sterling et E.Shapiro.

Si l'idée qui soutent Prolog est très prometteuse, l'utilisation de ce dernier pèche par un certain manque de performance.

Face à ce défaut, il est intéressant de recourir à une technique d'analyse statique - "l'interprétation abstraite".

C'est pourquoi, dans ce travail, nous exposerons le programme que nous avons élaboré et qui utilise ce concept via une méthode tabulée d'interprétation.

Nous essaierons, dans un premier temps, d'appréhender l'interprétation abstraite tabulée d'un point de vue théorique. Ensuite nous présenterons l'implémentation d'un programme utilisant cette technique.

Nous commencerons par un rappel de quelques bases théoriques ainsi qu'une présentation de l'analyse statique et de l'interprétation abstraite.

Le reste du travail se décomposera en deux parties que nous avons nettement séparées.

La première partie traitera de l'interprétation tabulée. Pour ce faire, nous présenterons d'abord la sémantique apportée par cette méthode. Ceci nous amènera à la découverte d'un algorithme permettant ce type d'interprétation (algorithme développé par le professeur Filé).
Nous terminerons ce chapitre par un exemple illustrant l'utilisation de l'algorithme.

La seconde partie, quant à elle, présentera notre implémentation de l'interpréteur tabulé ainsi qu'une implémentation du domaine abstrait PROP, que nous utiliserons avec l'interpréteur. Après une rapide présentation du fonctionnement de notre programme, nous verrons des exemples de résultats fournis par l'application.

RAPPELS

II. Entrée en matière

A. Rappels Mathématiques

1. Ordre partiel Complet

Soit S un ensemble. Une *relation* R sur S est un sous-ensemble de $S \times S$
Nous noterons le fait que $(x,y) \in R$ par xRy.

Un ***ordre partiel*** sur un ensemble S est une relation notée $\leq$ telle que:

1. $x \leq x \quad \forall x \in S$
2. $x \leq y \ \wedge \ y \leq x \ \Rightarrow \ x = y \quad \forall x, y \in S$
3. $x \leq y \ \wedge \ y \leq z \ \Rightarrow \ x \leq z \quad \forall x, y, z \in S$

Soit S un ensemble muni de la relation d'ordre partiel $\leq$.
$a \in S$ est une ***borne supérieure*** d'un sous-ensemble $X \subseteq S$, si $x \leq a \ \forall x \in X$.
$b \in S$ est une ***borne inférieure*** de X si $b \leq a \ \forall x \in X$

Soit S un ensemble muni de la relation d'ordre partiel $\leq$.
$a \in S$ est la ***plus petite borne supérieure***, notée ***lub*** (pour least upper bound), d'un sous-ensemble $X \subseteq S$ si a est une borne supérieure de X et si pour toute borne supérieure $a' \in X$, $a \leq a'$

On définit de la même manière une ***plus grande borne inférieure***, notée ***glb*** (pour greatest lower bound). Si elles existent, ces *lub* et *glb* sont uniques.

Une ***chaîne*** de $(S, \leq)$ est une séquence croissante $x_0 \leq x_1 \leq \ldots \leq x_n \leq \ldots$

On dit que $(S,\leq)$ est un ***ordre partiel complet***, noté ***cpo*** (pour complete partial order), s'il existe un plus petit élément nommé *bottom* et noté $\perp \in S$
Toutes les chaînes de S ont une lub notée $\bigcup^{\infty}_{n=0} x_n$

2. Treillis

Un treillis est un ensemble partiellement ordonné $(S,\leq)$ dans lequel toute paire d'élément s_1 et s_2 à un lub et un glb dans S

Un treillis $(T,\leq)$ est un treillis complet si lub(X) et glb(X) existent pour tout sous-ensemble X de T.

Nous avons ainsi :

$$\text{lub}(T) = \top$$
$$\text{glb}(T) = \bot$$

Cela signifie que deux éléments du treillis ont toujours un ancêtre commun et un descendant commun.

3. Monotonie

Soient T_1 et T_2 des treillis complets, et F une application de F: $T_1 \rightarrow T_2$
On dit que F est monotone si et seulement si

$$x \leq y \text{ dans } T_1 \Rightarrow F(x) \leq F(y) \text{ dans } T_2$$

4. Continuité

Soit T un treillis complet, et F une application F: $T \rightarrow T$
On dit que F est continue si pour toute chaîne $(x_n)_{n\geq 0}$, nous avons:

$$F(U^{\infty}_{n=0}\, x_n) = U^{\infty}_{n=0}\, F(x_n)$$

B. Analyse Statique

Il existe deux moyens de découvrir les propriétés d'un programme. Le premier est tout simplement d'exécuter ce programme: on presse la touche RETURN et on observe ce qui se passe à l'écran. L'alternative est de s'attabler, et d'analyser le texte source du programme pour le comprendre.

C'est à cette seconde méthode que s'attache le terme d'analyse statique (par opposition à la dynamique de l'exécution).

L'utilisation de l'analyse statique est soutenue par deux idées. Si ce que l'on cherche est formalisable, le processus d'analyse est fortement automatisable. De plus, la plupart du temps, seul ce type d'analyse est applicable à la recherche d'informations très particulières ou cachées.

Imaginons que nous ayons à compiler un programme; il convient d'utiliser la mémoire correctement et de prévoir si telle procédure retourne un entier signé ou pas. Ne pas s'adapter au comportement exact du programme pourrait avoir des effets néfastes selon les cas. Toujours en compilation, l'analyse d'un programme nous permettrait d'optimiser celui-ci: utiliser directement les registres du micro-processeur là où c'est possible, améliorer le comportement des boucles, ...etc...

Inutile de continuer à décrire les possibilités, celles-ci sont innombrables, et chacun est à même d'appréhender les utilisations possibles de cette méthode.

Il est cependant un domaine où l'interprétation statique a énormément d'importance. Celui des programmes logiques.

La programmation logique est basée sur la théorie des prédicats du premier ordre. Les objets ne sont plus définis en fonction d'un comportement à suivre, mais en fonction de leurs propriétés.
Par exemple, si l'on désire créer une procédure de concaténation de chaîne, telle qu'une chaîne de caractère soit la réunion de deux autres, on utilisera la propriéte TEXT3 = TEXT1 + TEXT2.

Le principal problème que pose la programmation logique, est sa caractéristique de *multidirectionalité*: un même paramètre d'une procédure, peut, suivant les circonstances, servir en tant que donnée ou en tant que résultat.
Reprenons l'exemple de la procédure APPEND(TEXT1, TEXT2, TEXT3); cette procédure peut servir aussi bien à trouver TEXT3 en connaissant TEXT2 et TEXT1, qu'à rechercher la chaîne de caractères (TEXT1) telle qu'elle se trouve être l'entête d'une chaîne connue (TEXT3) dont on connaît aussi la partie postérieure (TEXT2); ou encore rechercher toutes les chaînes possibles et imaginables telles que la troisième est somme des deux autres.

La multidirectionalité est un concept important, porte ouverte à l'intelligence artificielle. Hélas, le niveau de généralisation qu'elle implique nuit à son efficacité. Cependant, il existe des cas où des variables ne sont utilisées que d'une certaine manière. Il convient de détecter ces cas, et de concevoir un code plus adapté et plus efficace.

C. Interprétation Abstraite

Une difficulté vient de la sémantique complexe de Prolog. Puisque nous cherchons quelques propriétés particulières d'un programme, n'est-il pas possible de restreindre la sémantique utilisée, pour n'en garder qu'un sous-ensemble plus facile à gérer?
Là se trouve le point central de l'interprétation abstraite. Nous ne chercherons plus à travailler à l'aide de la sémantique normale mais sur une approximation de celle-ci.
Le domaine envisagé étant simplifié, les propriétés en seront d'autant plus faciles à déduire.

Pour bien comprendre ce concept, nous allons utiliser un exemple.
Nous avons à notre disposition une procédure qui travaille dans le domaine des entiers, et qui arrive à calculer le produit de deux nombres. Quand pourrons-nous utiliser un simple entier non signé comme résultat, quand devrons-nous utiliser un entier signé? Pour le savoir, il ne nous reste plus qu'à analyser. Hélas, il existe beaucoup d'entiers, et calculer tous les résultats de toutes les combinaisons possibles d'entiers est un travail titanesque.
On peut alors décomposer notre domaine des entiers selon trois qualités : les entiers positifs, le(s) entier(s) nul(s), les entiers négatifs. Selon cette décomposition, la réponse à notre question est quasi immédiate. En effet, le nombre de combinaisons possibles ne se chiffre plus qu'à neuf:

MULTIPLICATION	*POS*	*NUL*	*NEG*
POS	**POS**	**NUL**	**NEG**
NUL	**NUL**	**NUL**	**NUL**
NEG	**NEG**	**NUL**	**POS**

1. Domaine Abstrait

L'exemple précédent exprime clairement et simplement l'idée de l'interprétation abstraite.
Plutôt que de travailler directement avec un domaine de base (que nous appellerons **domaine concret**), nous allons essayer d'en délimiter les propriétés. Nous allons créer un autre domaine qui généralise le domaine concret. Ce nouveau domaine est appelé **domaine abstrait** car la généralisation engendre une abstraction du domaine concret.

Ce domaine abstrait ne peut être choisi au hasard. Il faut nécessairement que les propriétés du domaine concret -que l'on cherche à analyser- puissent se retrouver et soient conservées dans le domaine abstrait. Aussi, puisque le principe sera d'exécuter le programme, non plus sur le domaine concret, mais bien sur le domaine abstrait, il est essentiel que "les calculs sur le domaine abstrait puissent être réalisés de manière suffisamment efficace et convergent en un temps fini" [LECHAR92].

Pour utiliser un domaine abstrait, il faut tout d'abord édifier la relation entre les deux domaines. Ensuite, il faut créer, pour chaque opération du domaine concret, une abstraction de l'opération concrète qui travaillera sur le domaine abstrait.

Soit un domaine concret C et un domaine abstrait A. Pour être utilisable, A doit comprendre moins d'éléments que C. Mais aussi, pour chaque élément $c \in C$, il doit pouvoir exister un élément de $a \in A$, tel que a est une approximation (une abstraction) de c.
En "pratique", il suffit que A soit un treillis complet et qu'il existe deux fonctions
$\alpha: C \rightarrow A$ et $\gamma: A \rightarrow C$ où α forme avec γ une insertion de Galois:

1. $\forall c \in C : \gamma(\alpha(c)) \supseteq c$;
2. $\forall a \in A : \alpha(\gamma(a)) = a$;

α est appellée fonction d'abstraction, et γ est appellée fonction de concrétisation.

a) Opérations Abstraites

Notre but étant l'interprétation de programmes logiques, nous allons énumérer les fonctions nécessaires à ce cadre-là.
Le fonctionnement est basé sur l'utilisation et le traitement de substitutions. D'une part, il nous faut arriver à traiter l'unification. Et d'autre part, il faut aussi gérer les opérations de traitement des clauses lors de l'unification.

On peut définir la sémantique abstraite en terme de tuples abstraits $\{\theta_{in}, p, \theta_{out})$, où *p* est un symbole prédicatif d'arité *n*, et θ_{in} θ_{out} des substitutions abstraites sur les variables $(X_1,...,X_n)$. L'appel au programme *p* avec la substitution θ_{in} fournira une nouvelle substitution abstraite θ_{out}.

(1) Restriction

(a) RESTRC(c,θ)

On va restreindre une substitution θ aux seules variables présentes dans l'entête de la clause c.

(b) RESTRG(g,θ)

On va restreindre la substitution θ aux variables contenues dans le but g.

(2) Extension

(a) EXTC(c,θ)

On va étendre à l'ensemble des variables de la clause c une substitution limitée aux variables de l'entête de la clause c.

(b) EXTG(g,θ, θ')

θ' est une substitution limitée aux variables contenues dans le but g. Nous allons étendre θ' aux variables de θ.

(3) Unification

(a) AI_VAR(θ)

Cette fonction produit la substitution obtenue par l'unification entre deux variables.

(b) AI_FUNC(β, f)

Nous avons une unification de la forme var = foncteur(var_2,...,var_n). La fonction produit la valeur de la substitution définie sur l'ensemble des variables var_2,...,var_n.

(4) Union

(a) UNION(θ,θ')

Cette opération sert seulement à fournir la concaténation de plusieurs substitutions.

2. Exemple de domaine

a) PROP

Nous allons analyser ici le *groundness* d'une variable. Ground est un des modes possibles pour un terme dans un programme logique.
Selon Lloyd, "un terme *ground* est un terme ne contenant pas de variables. idem pour un atome" (in [LLOYD87]).

Plus pratique pour notre domaine, est la définition de Cortesi dans [COFIWI90]
"Un ensemble Σ de substitutions 'ground' x si chaque substitution de Σ instantie x à un terme ne comportant pas de variables"
ex: $\sigma = \{v_1/t_1, ..., v_n/t_n\}$ sera un substitution ground si tous les t_i sont des termes ground

PROP est un domaine abstrait tout particulièrement désigné pour représenter cette caractéristique de groundness.

(1) Substitutions abstraites

Cette représentation se fait au moyen de formules booléennes utilisant des connecteurs logiques $\wedge$, $\vee$, $\Leftrightarrow$ et $\neg$.

Intuitivement, la formule $x \wedge y$ signifie que les variables x et y sont ground. De même, la formule $x \wedge (y \Leftrightarrow z)$ signifie que x est ground si et seulement si y et z sont ground.

Soit D l'ensemble des variables $\{x_1, x_2, ..., x_n\}$ sur lequel est défini une substitution concrète.
Soit PF l'ensemble de toutes les formules propositionnelles sur D et les connecteurs logiques $\wedge$, $\vee$, $\Leftrightarrow$ et $\neg$.

(2) Propriétés

Une ***assignation de vérité*** est une fonction r: $D \rightarrow Bool$.
Une assignation de vérité r *vérifie* un ensemble de formules propositionnelles S, noté r(S), si r rend vrai chaque formule de S. On dit que, pour une variable x_i, $r(x_i)=true \Leftrightarrow \theta$ grounds $x_i \; \forall \; i \in \{1...n\}$.

L'évaluation d'une fonction f subordonnée à une variable x_i, est la fonction obtenue en remplacant x_i dans f par une une valeur booléenne b, ce que l'on note $f_{|xi=b}$.

Deux formules f_1 et f_2 de PF sont équivalentes, noté $f_1 \equiv f_2$, quand pour chaque assignation de vérité r, $r(f_1) = r(f_2)$.

(3) Restriction

Il faut éliminer de la formule toutes les variables qui n'appartiennent pas à l'entête. Soient $\{x_{n+1}, ..., x_m\}$ les variables qui n'apparaisssent que dans le corps.

RESTRC(c,f) = elim_all $[x_{n+1}, ..., x_m]$ f

avec elim_all défini récursivement comme ceci:

elim_all[] f = f

elim_all$[x_j,...,x_m]$ f = elim_all$[x_{j+1}, ..., x_m]$ $(f_{|xj=true} \vee f_{|xj=false})$ $(n<j\leq m)$

(4) Extension

Opération la plus facile:

EXTC(c,f) = f

(5) Unification

l'unification de deux variables ajoute simplement une équivalence à la présente substitution.

AI_VAR(f) = $f \wedge (x_1 \Leftrightarrow x_2)$

l'unification d'une variable x_1 et d'un foncteur $t(x_2,...,x_n)$ ajoute l'équivalence entre x_1 et la conjonction des variables contenues dans le foncteur.

AI_FUNC(f,t) = $f \wedge (x_1 \Leftrightarrow x_2 \wedge ... \wedge x_n)$

(6) Union

Consiste en la disjonction des différentes fonctions.

UNION$(f_1, ..., f_n) = f_1 \vee ... \vee f_n$

PARTIE I

III. Aspects théoriques

A. Modèle d'interprétation Tabulée

Le concept central de l'interprétation tabulée, est la conservation de l'ensemble des tuples $(\theta_{in},p,\theta_{out})$ pour l'utilisation des résultats θ_{out} par d'autres prédicats . L'utilisation d'une tabulation généralise la sémantique abstraite par rapport à celle que nous avons présentée au chapitre précédent.
Nous allons donc commencer par présenter cette nouvelle sémantique, avec tous les nouveaux termes qu'elle apporte.
Nous en arriverons ensuite à la description d'un algorithme mis au point par Codognet et Filé [COFILE91] et utilisant ce concept.

1. Systèmes de Calculs

Soient Σ l'ensemble des fonctions et prédicats, ainsi que ϑ l'ensemble dénombrable des variables.
Un **système de calcul** est un quintuple $C = (\Sigma', I, D, \oplus, \pi)$ où

- Σ' est un sous-ensemble fini de Σ,
- I est une interprétation des symboles de Σ',
- D est appelé le domaine de calcul, il est un sous-ensemble de toutes les formules bien formées sur Σ et ϑ,
- $\oplus$: $C\times C\rightarrow C$, est un opérateur associatif appelé **composition**,
- π: $C\times 2^{\vartheta}\rightarrow C$, une fonction appelée **projection**. Cette fonction nous sert à exprimer un élément du domaine D en fonction d'un certain nombre de variables. Supposons que nous ayons une contrainte $c \in D$, et V l'ensemble des variable de c:

  $$\pi(c, \{x_1, ..., x_n\}) = \exists(V/\{x_1, ..., x_n\}) . c$$

  exemple : Imaginons que nous travaillons avec le domaine de calcul abstrait PROP, et que {a, b, c, d, e} soient des éléments de ϑ.
  Si nous voulons calculer $\pi((a\wedge b\wedge(c\Leftrightarrow d))\vee d, \{b,c,d,e\})$, il nous faut utiliser la formule

  $$\exists a.((a\wedge b\wedge(c\Leftrightarrow d))\vee d).$$

  On peut dériver cette formule, pour n'utiliser que les opérateurs utilisés (et acceptés) dans PROP (en l'occurrence $\vee$, $\wedge$, $\Leftrightarrow$). Il nous faut considérer toutes les valeurs possibles de la fonction suivant les différentes valeurs que peut prendre a (dans ce cas-ci).

  $$\equiv ((TRUE\wedge b\wedge(c\Leftrightarrow d))\vee d) \vee (FALSE\wedge b\wedge(c\Leftrightarrow d)\vee d)$$
  $$\equiv ((b\wedge(c\Leftrightarrow d))\vee d) \vee d$$
  $$\equiv (b\wedge(c\Leftrightarrow d))\vee d$$

a) Programmes et calculs

Un **programme** portant sur un système de calcul *C*, est un ensemble de clauses de la forme $p(X) \leftarrow c, q_1(Y_1), ..., q_n(Y_n)$ où $X, Y_i \in \vartheta$; $c \in D$ et $p, q_i \in \Sigma$.

p(X) est appelé entête de la clause, le reste constituant le corps. Le corps de la clause, débute par la contrainte *c* suivie par l'ensemble de ce que nous appellerons *atome* $q_i(Y_i)$.

Un fait est une clause où n=0. Un **but** est une clause sans entête. Un but vide est un but où n=0.

Le **calcul** (ou *C*-calcul ou encore l'exécution) d'un *programme P avec un but B* est une séquence de but $<B_0, B_1, ..., B_n, ...>$*

où B_0 correspond au but B initial,

et si $B_i = \leftarrow c, A_0(X_0), A_1(X_1), ..., A_k(X_k)$

Alors

1- Il existe une clause $A_0 \leftarrow c_1, B_1(Y_1), ..., B_n(Y_n)$ qui est une variante d'une clause de P

2- soit m l'arité de A_0 et soit $c' = c \oplus c_1 \oplus (X_0^1 = Y_0^1) \oplus ... \oplus (X_0^m = Y_0^m)$** et $c' \neq$ FALSE

3- $B_{i+1} = \leftarrow c', B_1(Y_1), ..., B_n(Y_n), A_1(X_1), ..., A_k(X_k)$

Cette suite de buts correspond à l'ensemble des unifications qui ont lieu comme décrit dans la définition. A savoir que pour un but donné, on regarde s'il existe une clause H dans le programme P telle qu'avec une renomination des variables de H, cette clause puisse s'unifier. Pour que l'unification soit possible, il faut non seulement que les termes et arités de H et de A_0 soient identiques (point 1. de la définition), mais aussi et surtout que les contraintes de H et de B_i soient compatibles (point 2.). Si elles sont compatibles, la composition fournira une valeur non nulle, et on peut construire le but B_{i+1} présenté au point 3. Si elles ne sont pas compatibles, c' sera nul (FALSE) et dans ce cas B_{i+1} sera défini par $B_{i+1} = \leftarrow c' = \{FALSE\}$.

La définition d'un C-calcul n'est pas due au hasard. Au contraire, elle a été choisie parce qu'elle fait le lien avec le moteur d'inférence de Prolog. En effet, le but étant d'interpréter un programme Prolog, il convient que le fonctionnement du calcul soit aussi proche que celui de Prolog, autrement dit le calcul par dérivation-SLD.

---

* Cette séquence de but peut éventuellement être infinie (voir un peu plus bas)

** $X_0^j = Y_0^j$ représente l'opération de substitution de la $j^{ième}$ variables du vecteur X_0 par la $j^{ième}$ du vecteur Y_0

b) Simulation entre Systèmes de calculs

Supposons que nous ayons deux systèmes de calculs C et C'.
Dire que C' simule C étant donné une fonction de concrétisation γ: $C' \rightarrow 2^C$, est équivalent au fait que:

1. γ est une fonction monotone,
2. $\forall\, c \in C, \exists\, c' \in C' : c \in \gamma(c')$
3. $\forall\, c'_1, c'_2 \in C', \forall\, c_1, c_2 \in C : c_1 \in \gamma(c'_1), c_2 \in \gamma(c'_2) \Rightarrow c_1 \oplus c_2 \in \gamma(c'_1 \oplus c'_2)$

Il est évidemment possible d'avoir des simulations à plusieurs niveaux. Supposons qu'un système C'' simule un système C', et qu'un système C' simule un système C. Nous avons donc deux fonctions de concrétisation (γ et γ'), dont chacune respecte les conditions précédentes (en particulier la deuxième : $c \in \gamma(c')$; $c' \in \gamma'(c'')$). $X_0^j = Y_0^j$ représente l'opération de substitution de la j^{ieme} variables du vecteur X_0 par la j^{ieme} du vecteur Y_0

c) Programmes abstraits

Supposons que nous ayons un domaine C' qui simule un autre domaine C suivant la définition donnée dans le paragraphe précédent.

Si nous avons un but B à résoudre avec son programme P correspondant, il nous est possible de construire un programme P' ainsi qu'un but B' de telle manière que le C'-calcul de *P' avec B'* simule le C-calcul de *P avec B*.

Le programme P' est, comme défini au paragraphe , un ensemble de clauses de la forme:

$H(Y_0) \leftarrow c', B_1(Y_1), \ldots, B_n(Y_n)$.

Chaque clause de P' correspond en fait aux clauses $H(Y_0) \leftarrow c, B_1(Y_1), \ldots, B_n(Y_n)$ de P où $c \in \gamma(c')$.

Le but B' se défini de manière analogue à partir du but B.

exemple: soit le programme APPEND
en PROLOG

```
APPEND([], X, X).
APPEND([H|X], Y, [H|Z]) :- APPEND(X, Y, Z).
```

On peut réécrire ce programme pour qu'il corresponde à la définition de la page 12

```
APPEND(X1,X2,X3):- {X1/[], X2/X3}.
APPEND(X1,X2,X3):- {X1/[X4|X5],X3/[X4|X6]},APPEND(X5,X2,X6).
```

Si nous désirons travailler avec PROP (voir chapitre II.2) comme domaine abstrait de calcul, il nous est possible de définir le programme APPEND' comme suit:

```
APPEND(X1,X2,X3):-{X1∧(X2⇔X3}.
APPEND(X1,X2,X3):-{(X1⇔(X4∧X5))∧(X3⇔(X4∧
X6))},APPEND(X5,X2,X6).
```

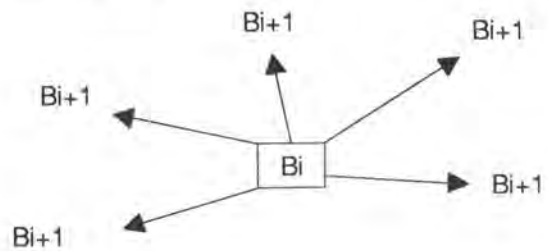
La contrainte associée à la première clause signifie que le paramètre X1 est ground et X2 est ground si et seulement si X3 est ground.

Quand à la seconde clause, la contrainte exprime que X1 est ground ssi X4 et X5 sont ground, ainsi que X3 est ground ssi X4 et X6 sont ground.

2. L'Interpréteur Tabulé

Le C-calcul comme défini précédemment, fournit UNE solution à un but donné.

En réalité, l'exécution d'un programme correspond à plusieurs C-calcul. En effet, le passage d'un but B_i à un but B_{i+1} se fait par une unification avec une clause du programme. Or, il existe sans doute plusieurs clauses avec lesquelles ce dernier puisse s'unifier. Cela signifie que la résolution n'est pas linéaire, mais rayonnonnante.



L'interprétation consiste à recueillir l'ensemble des solutions.

a) Définitions

Pour un atome A du corps d'une clause, la notation $\mathbf{Var_{cl}(A)}$ représente l'ensemble des variables contenues dans la clause. La même notation est utilisée pour les variables d'un but.

Toujours pour un atome A, la notation **Var(A)** représente uniquement les variables contenues dans cet atome.

Un **call-pattern** (littéral d'appel) est une paire de la forme [A,c] où A est un atome et c un élément du domaine de calcul considéré.

On appelle **left-most call-pattern** [*lf(B)*] d'un but $B = \leftarrow c, A_1,...,A_n$
le littéral d'appel $[A_1, \pi(c, Var_{cl}(A_1))]$.

L'ensemble *CP(B)*de call-pattern de B est défini récursivement:

- si B est un but vide, alors $CP(B) = \varnothing$
- sinon B est de la forme $\leftarrow c, A_1,A_2,...,A_n$, et $CP(B) = \{lf(B) \cup CP(B')\}$

où $B' = \leftarrow c,A_2,...,A_n$

Soient les deux littéraux d'appel [A,c] et [A',c'] de *C*.
Supposons que A est $p(X_1,...,X_n)$ et A' est $p(Y_1,...,Y_n)$,
et A=A' la représentation de $(X_1=Y_1 \oplus ... \oplus X_n=Y_n)$.
On dit que [A,c] et [A',c'] sont **équivalents** (A≡A'), si et seulement si $c \oplus (A=A') \equiv c' \oplus (A=A')$.

Un **call-exit marker** (marqueur de sortie) est un littéral d'appel particulier noté [A=H,c]•

Un **but généralisé**, est un but de la forme $\leftarrow c, A_1,...,A_k$ où A_i est soit un atome, soit un call-exit marker.

La signification d'un call-exit marker et d'un but est donnée un peu plus loin.

b) Résolution d'un goal généralisé

Supposons que nous ayons un but $B :: \leftarrow c, A_1,...,A_n$ à résoudre.
Pour exécuter cette résolution, on procède comme expliqué précédemment dans le paragraphe *programmes et calculs (p. 13)*.

On examine le littéral d'appel du but à résoudre. Face à ce call-pattern constitué du prédicat A_1 et de sa substitution d'entrée c, deux cas se présentent à nous: soit il s'agit d'un call-pattern que nous avons déjà rencontré, soit nous ne l'avons jamais vu et il nous faut l'examiner, le calculer.

(1) Résolution d'un prédicat non-vu

Nous cherchons à calculer $B_2::\leftarrow c', A_2, ..., A_n$, où "c' " est une nouvelle substitution.
$[A_1,c]$ nous est inconnu, il nous faut donc le résoudre. On vérifie s'il n'existe pas, dans le programme, une clause qui puisse s'unifier avec notre atome. Le cas échéant, il nous faut résoudre $[A_1,c]$ en faisant l'unification avec la clause. Cela revient à dire que nous allons calculer une nouvelle substitution "c' " correspondante.

Supposons qu'il existe dans le programme une clause
$H \leftarrow ch, K_1,..., K_m$ telle que A_1 puisse s'unifier avec H,

nous obtenons alors un nouveau but
$B_2::\leftarrow c', K_1,...,K_m, A_2,...,A_n$; où $c' = c \oplus ch \oplus (A_1=H)$

S'il est impossible à notre atome A_1 de s'unifier, alors notre but B est insolvable et les solutions sont la réponse FALSE (c'est-à-dire aucune substitution possible).

(2) Résolution d'un prédicat analogue à un autre déjà résolu

Soit le call-pattern $[A_1,c]$ équivalent à un autre que nous avons déjà rencontré, par exemple [H, c'].
On peut tenir le raisonnement suivant: si on devait résoudre notre call-pattern $[A_1,c]$, on retrouverait les mêmes solutions que celles trouvées lors de la résolution de [H,c'] (en tenant compte des renominations de variables).
Par conséquent, il semble inutile de procéder à nouveau au calcul des solutions de $[A_1,c]$; il suffit de reprendre directement celles de [H,c'] et de les intégrer dans notre but B pour obtenir un nouveau but $B_2::\leftarrow c', A_2, ..., A_n$ tel que $c' = c \oplus (A_1=H)$.

Réutiliser des solutions est une étape capitale dans notre raisonnement, car non seulement on évite de recalculer la valeur de la substitution (épargnant par là-même des ressources-machines telles que le temps et la mémoire), mais cela permet surtout de contourner le problème de la récursivité, c'est-à-dire le problème de la "non-finitude" de la résolution.

(3) Prise en compte de nouvelles solutions

A un moment de l'exécution, nous sommes arrivés à un but $B :: \leftarrow c, A_1,...,A_n$.
Celui-ci, avait donné, par substitution avec une clause H, le but $B_2::\leftarrow c',K_1,...,K_m, A_2,...,A_n$.
Après un certain nombre d'étapes, les atomes K_i ont été résolus, et nous nous sommes retrouvés avec un but $B_k::\leftarrow c'', A_2, ..., A_n$. Cela signifie que le call-pattern $[A_1,c]$, correspondant au lointain ancêtre but B, a été résolu; la substitution c'' en est la solution qu'il faut reporter sur les variables de B.
La solution de $[A_1,c] = c \oplus c'' \oplus (A1=H)$.

c) Algorithme d'Interprétation

A l'aide de ce que l'on vient de voir, on peut enfin se donner un algorithme qui nous permettra de calculer toutes les substitutions, solutions d'un but d'entrée donné.

Pour pouvoir facilement localiser un but correspondant à la résolution d'un atome ancêtre, nous allons travailler avec des **buts généralisés**.

Nous rappelons qu'un but généralisé est un but de la forme $\leftarrow c, A_1,...,A_k$ où A_i est soit un atome, soit un **call-exit marker** - c'est-à-dire un call-pattern spécial noté $[A=H,c]\bullet$.
Ce call-exit marker servira à rappeler la place, lors de l'exécution, où la clause H s'est substituée à l'atome A. Il permet de signaler que l'on est revenu de tous les appels récursifs qui auraient eu lieu pour résoudre l'atome A si nous avions travaillé dans un domaine concret. Lorsque ce marqueur arrive à la position extrême gauche d'un but, c'est qu'une solution à été apportée à l'atome A.

L'idée de l'algorithme est de conserver tous les call-pattern rencontrés lors des calculs, et donc, parallèlement, les solutions trouvées pour chaque call-pattern. Ainsi, lorsque plus tard, dans nos calculs, on se retrouvera avec un but dont le call-pattern est équivalent à un autre présent dans la table, on utilisera, comme présenté au paragraphe précédent, les solutions déjà trouvées pour composer une nouvelle substitution et créer le but suivant.

Nous allons aussi utiliser un arbre. Cet arbre représentera le parcours des différentes substitutions (parallèlement à une dérivation SLD), et conservera ainsi l'évolution du calcul, chaque noeud représentant en fait un but généralisé à résoudre.

A chaque étape, l'algorithme modifie l'arbre et la table en y ajoutant des éléments. Par mesure de facilité, nous noterons T_n et t_n, respectivement l'état de la table et de l'arbre à la $n^{ième}$ étape.

La séquence de paires arbres/tables $< (t_0,T_0), (t_1,T_1), ..., (t_n,T_n), ... >$ sera appelée **tabled computation** ou interprétation tabulée.

Voyons maintenant cet algorithme interprétation tabulée étant donné un *C-programme P et un C-but B*

(1) Algorithme

(a) Initialisation

T_0 est vide

t_0 est une simple racine étiquetée de B

(b) Itération - passage de (t_i, T_i) à (t_{i+1}, T_{i+1})

Choisissons un des noeuds-feuilles, que nous noterons *n*, de t_n.

Soit F::← c, A_1, ..., A_k le but qui étiquette cette feuille *n*, et F est un but non-vide.

Examinons A_1. Comme F est un but généralisé, A_1 peut être de deux types possibles, qu'il nous faut distinguer:

1. A_1 est un atome.

On regarde s'il n'existe pas dans T_i une entrée équivalente à $[A_1, \pi(c, Var(A_1))]$.

a). Si elle n'existe pas, alors nous dirons que *n* est un **noeud solution.** On crée t_{i+1} en modifiant t_i de telle manière que pour chaque clause $cl = H \leftarrow c', B_1, ..., B_m$ du programme P pour lequel $c \oplus c' \oplus (A_1 = H)$ est non nul (autrement dit qui puisse s'unifier), un nouveau noeud-fils *n'* est ajouté à *n* dans t_i. Ce nouveau noeud sera étiqueté par le but

$\leftarrow c'', B_1, ..., B_m, [A_1=H,c]\bullet, A_2, ..., A_k$

où $c'' = \pi(c \oplus c' \oplus (A_1=H), Var_{cl}(cl))$.

T_{i+1} est obtenu en ajoutant une nouvelle ligne de clé $[A_1, \pi(c, Var(A_1))]$ et une liste vide de solutions.

b) Il existe dans T_i une ligne *l* dont la clef [A', c'] est équivalente à $[A_1,\pi(c,Var(A_1))]$. Nous appellerons *n* un **noeud référence**, car une référence est instaurée entre *n* et la liste des solutions de *l*. On obtient t_{i+1} en ajoutant à *n*, pour chaque solution c'' présente dans la liste de solutions de *l*, un noeud-fils *n'* étiqueté par $\leftarrow \pi(c'' \oplus (A_1 = A'), Var_{cl}(A1)), A_2, ..., A_k$.

$T_{i+1} = T_i$ - en un mot, T est inchangé

2. A_1 est un call-exit marker [A=H,c']

Nous appellerons ce noeud *n* un **noeud spécial**.

Il existe, impérativement, dans T_i une ligne ayant une clef de valeur [A,c'].

A moins qu'il n'existe déjà une solution équivalente, on ajoute à cette ligne, la solution $c'' = \pi(c \oplus c' \oplus (A=H), Var_{cl}(A))$. On trouve ainsi T_{i+1}.

On ajoute à *n* un noeud-fils étiqueté par $\leftarrow c'', A_2, ..., A_k$

Il faut aussi, si on a introduit une solution à T_i, ajouter un noeud-fils supplémentaire à chaque noeud référence qui pointe vers la ligne de solution en question. Si $\leftarrow c_r, B_1, ..., B_p$ est l'étiquette du noeud référence considéré, on lui ajoute un nouveau fils étiquetté par $\leftarrow c_n, B_2, ..., B_p$

où $cn = \pi(c'' \oplus (A=B_1), Var_{cl}(B_1))$

(c) Terminaison

Dès qu'il n'existe plus que des noeuds étiquettés par un but vide, l'interprétation est terminée.

d) Condition d'équivalence

Le fait d'utiliser une table pour améliorer l'interprétation, est-il complètement équivalent à une exécution comme présentée au point III.A.1.a)? Ne risque-t-il pas d'y avoir des différences par rapport à une dérivation-SLD?

Considérons un calcul $<B_0, B_1, ..., B_k>$ où

$B_0 = \leftarrow c_0, A_1, A_2, ..., A_n$
$B_1 = \leftarrow c_1, P_1, ..., P_m, A_2, ..., A_n$
.
.
$B_k = \leftarrow c_k, A_2, ..., A_n$

La clause utilisée pour passer de B_0 à B_1 est $cl = H \leftarrow c', P_1, ..., P_m$

Soient $c'_0 = \pi(c_0, Var_{cl}(A_1))$ et $c'_k = \pi(c_k, Var_{cl}(cl))$. On peut dire que l'interprétation tabulée est équivalente si les deux points suivants sont vérifiés:

1) $\pi(c_1, Var_{cl}(cl)) \equiv \pi(c'_0 \oplus c' \oplus (A_1 = H), Var_{cl}(cl))$

2) $\pi(c_k, Var_{cl}(A_1)) \equiv \pi(c'_0 \oplus c' \oplus (A_1 = H), Var_{cl}(A_1))$

La première condition impose que les éléments de la contrainte c_0 provenant des atomes $A_2,..,A_n$ n'aient aucune influence sur les calculs de la nouvelle contrainte issue de l'unification entre A_1 et H.

La seconde condition demande que lorsqu'on trouve une solution à A_1, il soit possible de combiner cette solution aux éléments de la contrainte c_0 provenant des atomes $A_2,..,A_n$ pour obtenir la nouvelle contrainte c_k.

Si ces deux conditions sont vérifiées, nous sommes sûrs que lors de l'unification entre A_1 et H il n'est absolument pas nécessaire de conserver les parties de contraintes dont A_1 ne se "sert" pas, puisque il sera possible de les retrouver.

B. Exemple d'interprétation abstraite

Nous avons vu comment fonctionne l'interpréteur tabulé. Le fonctionnement a été décrit dans un formalisme mathématique, suivant une méthode d'induction.

Nous vous proposons, maintenant, d'en découvrir le fonctionnement dans son ensemble à l'aide d'un exemple.
Nous pourrons de la sorte voir la réelle dynamique de cet algorithme.

L'exemple que nous vous proposons est "append", en utilisant le domaine abstrait PROP. Le choix de ce programme est simple:

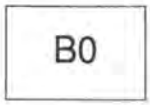
- le code du programme a déjà été maintes fois présenté, et il est donc inutile d'y revenir,
- son interprétation n'est pas trop longue,
- et surtout, on y voit toutes les phases de l'algorithme.

Fixons-nous comme départ le but suivant:

$$B_0 = \leftarrow \{A \wedge B\}, \text{append}(A,B,C).$$

Nous représenterons les arbres et les tables successifs par les noms A_i et T_i.
Pour faciliter la lecture, nous ne représenterons les noeuds que par un identifiant, et nous reprendrons sur le côté la liste des buts constituant les noeuds; de même, les contraintes recevront un nom, et seront exprimées à part.
Chaque fois que nous voudrons insérer un nouveau noeud dans l'arbre, ce noeud sera représenté par un rectangle. Après analyse, les "noeuds solutions" seront représentés par un cercle, les "noeuds référence" par un parallélogramme, les "noeuds spéciaux" par un hexagone et les "noeuds terminaux" par un ovale.
Les nouveaux éléments insérés dans la table seront écrits en gras.

1. Initialisation



A_0 =

$T_0 = \varnothing$

$B_0 = \leftarrow c_0$, append(A,B,C).
$c_0 = \{A \wedge B\}$

2. Itération

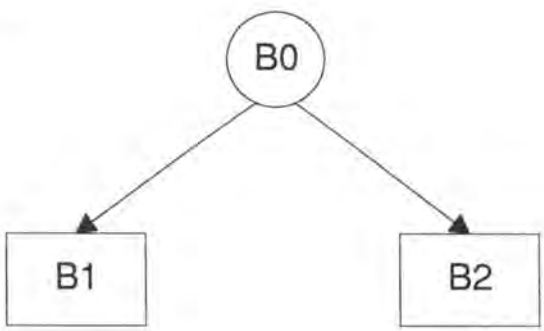
a) étape 1

On choisit un noeud qui n'a pas encore été analysé. Ici le choix est simple, il n'existe que le noeud B_0.
Le littéral le plus à gauche est [append(A,B,C), A∧B]. Puisque la table est vide, nous avons affaire à un noeud solution.

Ce noeud peut s'unifier (après renomination des variables) avec la première clause du programme H_1 = append(D,E,F)←{D∧(E⇔F)}. et avec la deuxième H_2 = append(G,H,I)←{(G⇔(J∧K))∧(I⇔(J∧L))}, append(K,H,L).

L'unification nous fournit donc deux nouveaux noeuds.

A_1 =



$B_0 = \leftarrow c_0$, append(A,B,C).
$B_1 = \leftarrow c_1, []_1$.
$B_2 = \leftarrow c_2$, append(K,H,L), $[]_2$.

$c_1 = \pi(c_0 \wedge (D \wedge (E \Leftrightarrow F)) \wedge (A \Leftrightarrow D) \wedge (B \Leftrightarrow E) \wedge (C \Leftrightarrow F), \{D\ E\ F\})$
$= \pi(A \wedge B \wedge C \wedge D \wedge E \wedge F, \{D\ E\ F\})$
$= D \wedge E \wedge F$

$c_2 = \pi(c_0 \wedge (G \Leftrightarrow (J \wedge K)) \wedge (I \Leftrightarrow (J \wedge L)) \wedge (A \Leftrightarrow G) \wedge (B \Leftrightarrow H) \wedge (C \Leftrightarrow I), \{G\ H\ I\ J\ K\ L\})$
$= \pi(A \wedge B \wedge G \wedge J \wedge K \wedge H \wedge I \Leftrightarrow (J \wedge L) \wedge (C \Leftrightarrow I), \{G\ H\ I\ J\ K\ L\})$
$= G \wedge J \wedge K \wedge H \wedge I \Leftrightarrow (J \wedge L) = G \wedge J \wedge K \wedge H \wedge (I \Leftrightarrow L)$

$[]_1$ = [append(A,B,C) = append(D,E,F), A ∧ B]
$[]_2$ = [append(A,B,C) = append(G,H,I), A ∧ B]

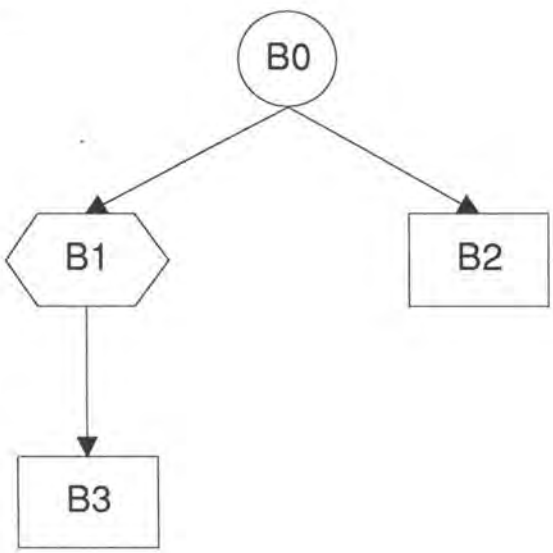
On crée une entrée [append(A,B,C), A $\wedge$ B] dans la table .

T_1	
littéraux d'appel	*liste de solutions*
[append(A,B,C), A $\wedge$ B]	

b) étape 2

Nous avons le choix entre deux noeuds: B_1 et B_2; traitons B_1.
Le littéral le plus à gauche est un call-exit marker.
On calcule la valeur abstraite $c_3 = \pi(c_1 \wedge c_0 \wedge (A \Leftrightarrow D) \wedge (B \Leftrightarrow E) \wedge (C \Leftrightarrow F), \{ A\ B\ C\})$
$= \pi(A \wedge B \wedge C \wedge D \wedge E \wedge F, \{A\ B\ C\}) = A \wedge B \wedge C.$
La ligne d'entrée [append(A,B,C), A $\wedge$ B] ne contient pas encore de solution, par conséquent, on lui insére c_3. Il n'existe pas de noeud qui fait référence à cette liste de solutions.
On crée alors le noeud B_3 fils de B_1.

$A_2 =$



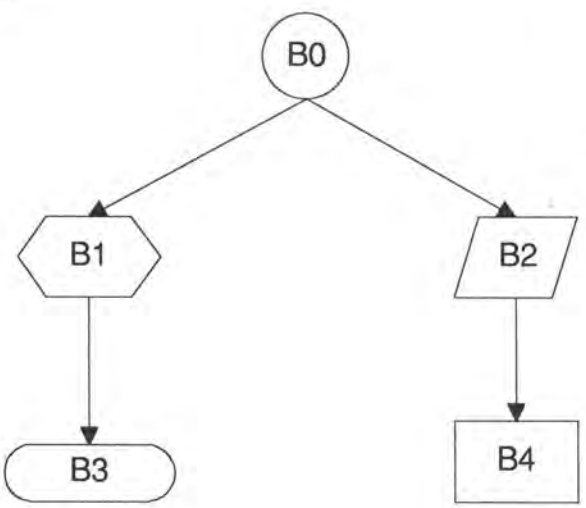
$B_0 = \leftarrow c_0$, append(A,B,C).
$B_1 = \leftarrow c_1, []_1$.
$B_2 = \leftarrow c_2$, append(K,H,L), $[]_2$.
$B_3 = \leftarrow c_3$.

T_2	
littéraux d'appel	*liste de solutions*
[append(A,B,C), A $\wedge$ B]	**A $\wedge$ B $\wedge$ C**

c) étape 3

Le noeud B_3 est un but vide. On choisit alors le but B_2, qui n'est pas vide et n'a pas encore été analysé.
Son littéral d'appel est [append(K,H,L), $\pi(c_2,\{K\ H\ L\} = K \wedge H]$. Il existe dans la table une ligne *l*=[append(A,B,C), $A \wedge B$]. Ces deux littéraux sont équivalents, donc B_2 est un noeud référence.
Comme la ligne *l* contient une solution, on crée le noeud B_4.

A_3 =



$B_0 = \leftarrow c_0$, append(A,B,C).
$B_1 = \leftarrow c_1, []_1$.
$B_2 = \leftarrow c_2$, append(K,H,L), $[]_2$.
$B_3 = \leftarrow c_3$.
$B_4 = \leftarrow c_4, []_2$.

$c_4 = \pi(c_3 \wedge (K \Leftrightarrow A) \wedge (H \Leftrightarrow B) \wedge (L \Leftrightarrow C), \{K\ H\ L\})$
$= K \wedge H \wedge L$

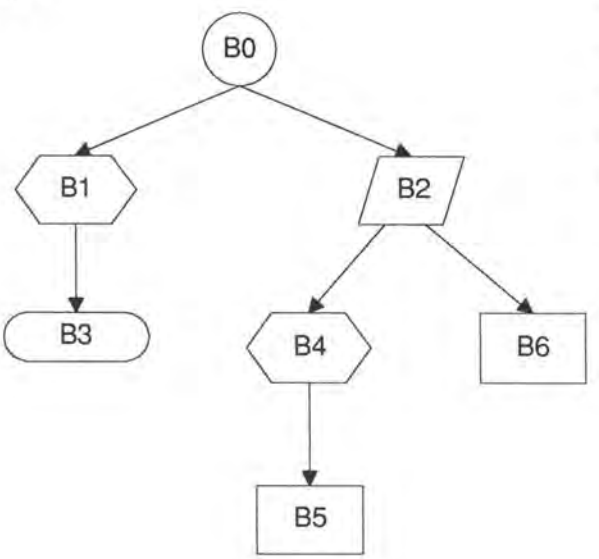
La table est inchangée: $T_3 = T_2$

T_3	
littéraux d'appel	*liste de solutions*
[append(A,B,C), $A \wedge B$]	$A \wedge B \wedge C$

d) étape 4

Le noeud B_4 est un call-exit marker [append(A,B,C) = append(G,H,I), $A \wedge B$].
Tout comme à l'étape 2, ce call-exit apporte une solution au littéral d'appel de la même ligne *l*.
On calcule $c_5 = \pi(c_4 \wedge (A \wedge B) \wedge (A \Leftrightarrow G) \wedge (B \Leftrightarrow H) \wedge (C \Leftrightarrow I), \{ A\ B\ C\})$
$= A \wedge B$.
On crée le noeud B_5.
On remarque qu'il n'existe pas encore de solution équivalente à c_5 dans la liste jointe à la ligne *l*, donc on l'y ajoute. Comme le noeud B_2 contient une référence à la ligne *l*, il nous faut étendre B_2 à la solution c_5 en créant le noeud B_6.

$A_4 =$



$B_0 = \leftarrow c_0$, append(A,B,C).
$B_1 = \leftarrow c_1, []_1$.
$B_2 = \leftarrow c_2$, append(K,H,L), $[]_2$.
$B_3 = \leftarrow c_3$.
$B_4 = \leftarrow c_4, []_2$.
$B_5 = \leftarrow c_5$.
$B_6 = \leftarrow c_6, []_2$.

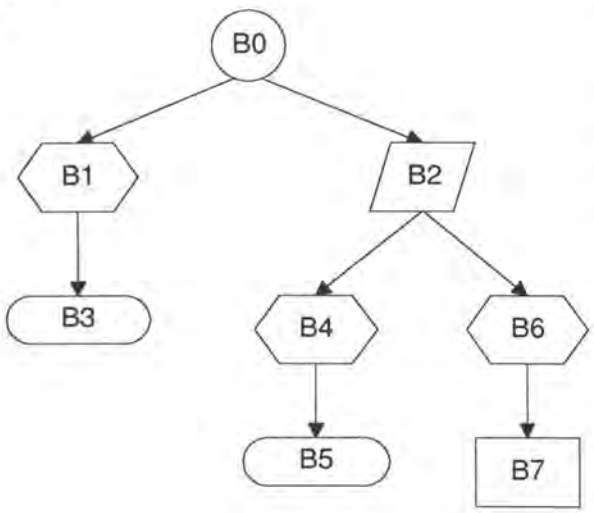
$c_5 = A \wedge B$
$c_6 = \pi(c_5 \wedge (A \Leftrightarrow K) \wedge (B \Leftrightarrow H) \wedge (C \Leftrightarrow L), \{K\ H\ L\})$
$= H \wedge L \wedge K$

T_4	
littéraux d'appel	*liste de solutions*
[append(A,B,C), $A \wedge B$]	$A \wedge B \wedge C$; **$A \wedge B$**

e) étape 5

Le noeud B_5 est étiqueté par un but vide.
On choisit alors B_6, dont le littéral le plus à gauche est un call-exit marker.
Ce marker correspond toujours à la même ligne *l* de la table.
On calcule $c_7 = \pi(c_6 \wedge (A \wedge B) \wedge (A \Leftrightarrow G) \wedge (B \Leftrightarrow H) \wedge (C \Leftrightarrow I), \{ A\ B\ C \})$
$= A \wedge B \wedge C$. Comme il existe déjà une solution équivalente dans la table, on ne crée que le noeud B_7

$A_5 =$



$B_0 = \leftarrow c_0$, append(A,B,C).
$B_1 = \leftarrow c_1, []_1$.
$B_2 = \leftarrow c_2$, append(K,H,L), $[]_2$.
$B_3 = \leftarrow c_3$.
$B_4 = \leftarrow c_4, []_2$.
$B_5 = \leftarrow c_5$.
$B_6 = \leftarrow c_6, []_2$.
$B_7 = \leftarrow c_7$.

$c_7 = A \wedge B \wedge C$

La table est inchangée.

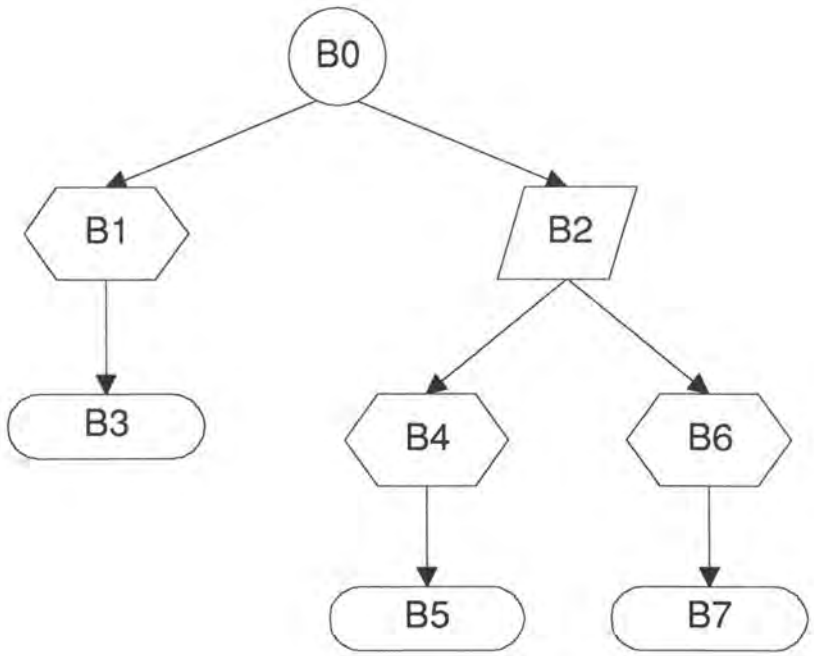
f) étape 6

Le noeud B_7 correspond à un but vide.
Il ne reste plus de noeud à analyser.

L'algorithme est terminé.

g) Résultat

Le résultat de l'interprétation de "append"par le but ← {A∧B}, append(A,B,C).
fournit l'arbre



et la table constituée d'une seule ligne

Table	
littéraux d'appel	*liste de solutions*
[append(A,B,C), A ∧ B]	A ∧ B ∧ C ; A ∧ B

PARTIE II

IV. Implémentation

Nous avons découvert un algorithme nous permettant d'interpréter un programme logique.
Cet algorithme, nous l'avons implémenté. Ce qui va suivre est la description de cette implémentation.

On y trouve, bien évidemment, la présentation de l'algorithme. Mais en plus, puisque nous désirions utiliser le domaine abstrait PROP, nous avons dû implémenter un système de gestion de formules propositionnelles.

Dans ce qui suit, on retrouve les deux facettes de cette description, en commençant par la gestion de formules booléennes.

Nous terminerons par une explication du fonctionnement du programme, ainsi que des exemples d'utilisations.

A. Gestion de fonctions booléennes

1. Présentation

Une même fonction booléenne peut être représentée de multiples manières.
En logique booléenne, on utilise plusieurs opérateurs :

- l'opérateur monadique de négativité $\neg$,
- l'opérateur de conjonction $\wedge$,
- l'opérateur de disjonction $\vee$,
- l'opérateur d'équivalence $\Leftrightarrow$,
- et enfin celui d'implication $\Rightarrow$.

On utilise aussi des quantificateurs d'existence $\exists$ et d'universalité $\forall$.

C'est grâce à des combinaisons d'opérateurs que l'on peut modifier la forme d'une fonction. Les méthodes de Carnaugh et les tables de vérités, qui sont les bases de l'enseignement de la logique des mathématiques, ont fait leurs preuves, et nous ont montré les équivalences entre fonctions. En l'occurrence, il nous est possible de représenter une équivalence par une disjonction de conjonctions.

$$\langle \alpha \Leftrightarrow \beta \rangle \Leftrightarrow \left\langle [\alpha \vee \beta] \wedge \left[\overline{\alpha} \vee \overline{\beta}\right] \right\rangle$$

Cette idée de représentation a été utilisée dans [NELISS91], permettant de faciliter la constitution des formules logiques. En effet, elle ne nécessite que deux éléments ($\wedge$ et $\vee$). L'idée géniale fut aussi de représenter une conjonction par une suite de bits, où un bit devient VRAI si et seulement si la variable d'indice correspondante à la position du bit est présente dans la conjonction. L'ensemble des conjonctions est simplement une liste chaînée de ces conjonctions.

Ce n'est pas cette méthode et son implémentation que nous allons présenter ici.
Nous allons au contraire nous baser sur les travaux de Bryant présenté dans [BRYANT86].
Le principe est d'utiliser une décomposition simple:

$$F(x_1,x_2,...,x_n) \Leftrightarrow F_{x1=vrai}(x_2,...,x_n) \vee F_{x1=faux}(x_2,...,x_n)$$

Puisque que ces deux fonctions sont équivalentes, on peut aisément représenter la fonction $F(x_1, x_2, ..., x_n)$ à l'aide d'un graphe dirigé acyclique.
Une fonction sera donc définie comme ceci:

Définitions[BRYANT86]:
1)Un graphe de fonction est un graphe dirigé ayant une racine, et composé d'ensemble de noeuds V de deux types.
Un noeud *nonterminal* v a comme attributs un argument indice *index(v)* $\in \{x_1, ..., x_n\}$ et deux fils *low(v)*, *high(v)* $\in$ V.
Un noeud *terminal* a un attribut de valeur *value(v)* $\in$ {TRUE, FALSE}.
De plus, pour tout noeud nonterminal, si low(v) est aussi nonterminal, alors index(v) < index(low(v)). Il en va de même pour high(v).

2) Un graphe de fonction représente une fonction F définie récursivement comme suit:
   a) Si v est un noeud terminal, alors F_v = TRUE si value(v) = TRUE, et sinon F_v = FALSE.
   b) Si v est nonterminal d'indice index(v) = x_i , alors F_v est la fonction

$$F_v(x_1, ..., x_n) = x_i \wedge F_{low(v)}(x_1, ..., x_n) \vee \neg x_i \wedge F_{high(v)}(x_1, ..., x_n)$$

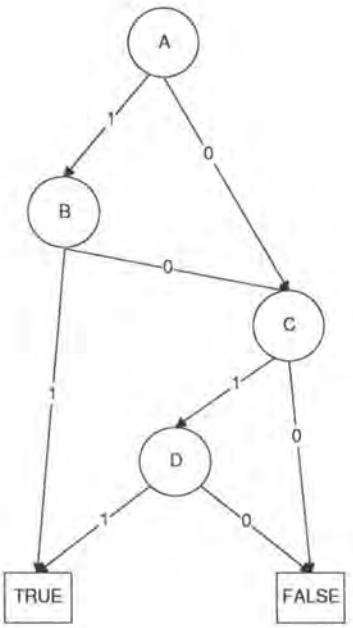
3) Deux graphes de fonctions G et G' sont isomorphique s'il existe une relation bijective σ des noeuds de G vers les noeuds de G' telle que pour chaque noeud n, si $\sigma(n) = n'$, alors

- soit n et n' sont terminaux et value(n) = value(n')
- soit n et n' sont nonterminaux et index(n) = index(n')
  et $\sigma(low(n)) = \sigma(low(n'))$
  et $\sigma(high(n)) = \sigma(high(n'))$

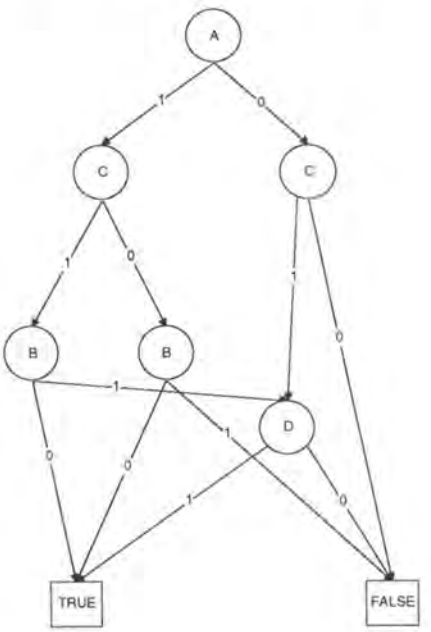
L'avantage non négligeable de cette représentation est l'existence d'une forme canonique grâce à l'ordonnancement (index(noeud-n) > index(noeud-fils-de-n)). De plus une fonction qui contient des sous-graphes isomorphiques peut facilement voir sa taille diminuer (voir fonction reduce).
Le seul inconvénient de cette représentation tient paradoxalement à son avantage: le **choix initial** de la méthode d'ordonnancement est primordiale (d'autant plus que la méthode doit être la même pour chaque fonction).

exemple: Imaginons que A,B, C, D soient des variables, et que nous voulions représenter la fonction $(A \wedge B) \vee (C \wedge D)$. Suivant la règle d'ordonnancement, la représentation sera différente, et par voie de fait, la taille.

Ordre : A < B < C < D



Si par contre nous utilisons comme ordre:
A < C < B < D on obtient:



Il est très difficile de savoir quelle est la meilleure méthode d'ordonnancement. Nous avons choisi la plus simple, qui est aussi celle préconisée par [BRYANT86], et qui consiste à utiliser l'indice de la variable: $x_1 < x_2 < ... < x_n$

Venons-en maintenant à l'implémentation.

Nous représenterons les sommets terminaux et nonterminaux par une seule et même structure que nous nommerons vertex:

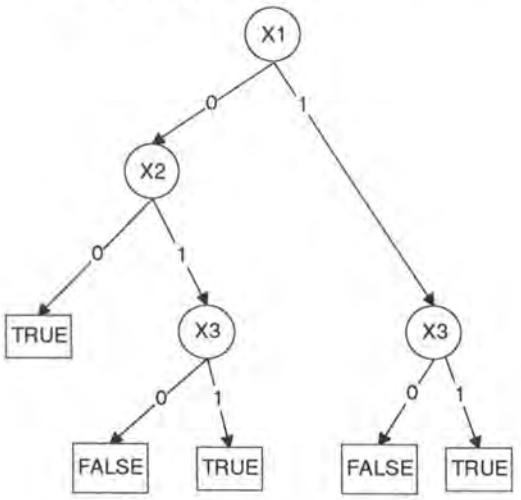
```
typedef struct tVertex
{
     PVertex low, high;
     int     index;
     int     value;
     int     id;
     BOOL    mark;
} Vertex;
```

"low" et "high" sont des pointeurs sur un autre vertex; ils sont nuls dans le cas d'un sommet terminal.
"index" est l'indice de la variable. Si le sommet est terminal, nous mettrons index à 0 (ce qui signifie que nous ne représenterons que des variables x_i pour $i \geq 1$.
"value" est la valeur des sommets terminaux(0 pour FALSE ou 1 pour TRUE); nous la mettrons à 2 dans le cas de sommets nonterminaux.
"id" sert à identifier chaque sommet au sein du graphe.
"mark" n'est qu'un indicateur utilisé lors de parcours récursif d'un graphe.

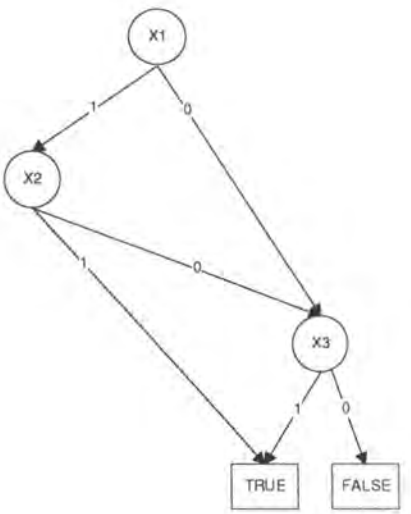
2. Fonctions associées

a) Réduction

Ce que nous cherchons à faire, est d'éliminer tous les sommets inutiles, tout en conservant une même valeur au graphe. C'est-à-dire que nous allons éliminer tous les sous-graphes isomorphiques (sauf un).
Imaginons que nous ayons un graphe représentant la fonction
$(x_1 \wedge x_2) \vee x_3$:



Et on voudrait obtenir ceci →



Ces deux graphes représentent bien la même fonction, néanmoins, le deuxième nécessite moins de sommets.

Pour éliminer les sous-graphes excédentaires, nous assignerons un identifiant à chaque sommet qui sera conservé dans le nouveau graphe, de telle manière que pour deux sommets u et v, $id(u) = id(v)$ si et seulement si $F_u = F_v$.
Nous fonctionnerons par induction sur l'index, en partant des sommets terminaux pour arriver à la racine.

- Nous ne conserverons que deux sommets terminaux: celui de valeur TRUE et le FALSE
- Supposons que nous ayons traité tous les sommets terminaux ainsi que tous ceux d'index > i

Un sommet s recevra un *id* égal à l'id d'un autre sommet ayant déjà été traité

si $id(low(s)) == id(high(s))$ car alors s est redondant [→action $id(s)=id(low(s))$

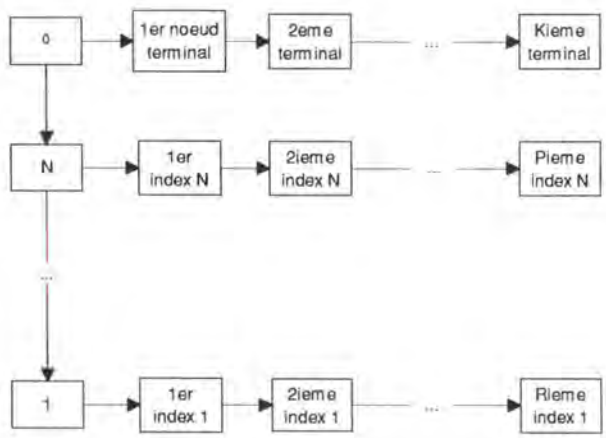
ou bien s'il existe un sommet u d'index i tel que $id(low(s)) == id(low(u))$ et $id(high(s))==id(high(u))$

[→ action $id(s)=id(u)$

sinon $id(s) = i$

algorithme

Nous allons arranger tous les sommets de même index dans une même liste. Puis créer une liste de ces listes de sommets, en commençant par la liste des sommets terminaux et en continuant par une liste d'index décroissante.




```
PFunction Reduce(f)
PFunction f;
{
     int nextid, keya, keyb, oldkeya, oldkeyb;
     PVertex u, r;
     PVList listofvertex;

     listofvertex = PutVertexinList(f);
     nextid = 0;

     while (listofvertex)
     {
          Emptyset();
          while (listofvertex->list)
          {
               u = listofvertex->list->elem;
               listofvertex->list = listofvertex->next;

               if (u->index == 0)
                    Addset(u->val, 0, u);
               else
                    if (u->low->id == u->high->id)
                         u->id = u->low->id;
                    else
                         Addset(u->low->id, u->high->id, u);
          }
          oldkeya = -1;
```

```
            oldkeyb = -1;
            while (Getset(&keya, &keyb, &u))
            {
                  if ((keya==oldkeya)&&(keyb==oldkeyb))
                        u->id = nextid;
                  else
                  {
                        nextid++;
                        u->id=nextid;
                        subgraph[nextid]=u;
                        if (u->val==2)
                        {
                              u->low = subgraph[u->low->id];
                              u->high = subgraph[u->high->id];
                        }
                        else
                        {
                              u->low = NULL;
                              u->high = NULL;
                        }
                        oldkeya = keya;
                        oldkeyb = keyb;
                  }
            }
            Deleteset();
            listofvertex = listofvertex->next;
      }
      f->root = subgraph[r->id];
      return (f);
}
```

b) Composition de deux fonctions

Pour mettre en oeuvre toutes les opérations décrites au début de ce chapitre, il conviendrait de créer une procédure par opération.
Fort heureusement, grâce à la représentation arborescente utilisée, il est possible d'implémenter tous les opérateurs diadiques à l'aide d'un seul et même algorithme.
L'idée maîtresse de cet algorithme, est la formule d'expansion de Shannon:

$$f_1 \otimes f_2 = \neg x_i \wedge (f_{1|xi=0} \otimes f_{2|xi=0}) \vee x_i \wedge (f_{1|xi=1} \otimes f_{2|xi=1})$$

Il est donc possible de créer $f_1 \otimes f_2$ récursivement en partant d'un sommet x_i.
Il suffit de prendre la racine de f_1 ou f_2 dont l'index est le plus petit (selon notre régle d'ordonnancement) comme x_i, il est possible d'appliquer cette formule* à la structure de nos arbres.
Il suffit de créer un nouveau sommet *ns* d'index i, et de calculer récursivement *low(ns)* et *high(ns)*.

---

* formule semblable à la représentation d'un fonction

$$F_v(x_1, ..., x_n) = x_i \wedge F_{low(v)}(x_1, ..., x_n) \vee \neg x_i \wedge F_{high(v)}(x_1, ..., x_n)$$

Il faut pourtant tenir compte de cas particuliers:
soient v_1 et v_2 les racines respectives de f_1 et f_2,

- si f_1 et f_2 correspondent à une fonction représentant la valeur TRUE ou FALSE (in extenso si les racines sont des noeuds terminaux), leur combinaison sera une fonction de valeur=value(v_1)⊗value(v_2),
- si v1 ou v2 (ou exclusif) est terminal, la combinaison sera fonction de l'opérateur (soit F la fonction représentée par l'arbre dont le sommet n'est pas terminal)

  ex: $F \wedge TRUE = F$
  $F \wedge FALSE = FALSE$
  $F \vee TRUE = TRUE$
  $F \vee FALSE = F$

Comme nous désirons utiliser un algorithme générique pour tous types d'opérateur, il nous faut trouver une technique pour détecter si la combinaison de deux sommets forme une valeur terminale (si c'est le cas, nous devrons arrêter la récursion dans cette branche de l'arbre). " On calcule *value(v_1)*⊗ *value(v_2)* en utilisant une logique à trois états {TRUE, FALSE, X}, ou X (la valeur de tous sommets nonterminaux) représente 'ne pas s'occuper'; telle que si b⊗TRUE = b⊗FALSE = a, alors b⊗X = a, sinon b⊗X = X."[BRYANT86]

Il nous suffit de représenter les opérateurs que nous désirons appliquer sur base de cette logique, et nous pourrons utiliser l'algorithme.

AND	*TRUE*	*FALSE*	*X*
TRUE	TRUE	FALSE	X
FALSE	FALSE	FALSE	FALSE
X	X	FALSE	X

OR	*TRUE*	*FALSE*	*X*
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	X
X	TRUE	X	X

EQU	*TRUE*	*FALSE*	*X*
TRUE	TRUE	FALSE	X
FALSE	FALSE	TRUE	X
X	X	X	X

Voici la procédure générique `APPLY_STEP`, qu'il suffit d'appeler avec comme arguments les pointeurs vers les sommets racines des fonctions, et un pointeur vers la fonction calculant la combinaison dans notre nouvelle logique.

```
PVertex Apply_Step(v1, v2, fct)
PVertex v1, v2;
int (*fct)(int, int);
{
      PVertex u, vlow1, vlow2, vhigh1, vhigh2;
      u = subgraph[T[v1->id][v2->id]];
      if (u!=NULL) return(u);

      u=NewVertex(1,pf);

      subgraph[indice] = u;
      T[v1->id][v2->id] = indice++;

      u->val = (*fct) (v1->val, v2->val);
      if (u->val!=2)
      {
            u->index=0;
            u->low=NULL;
            u->high=NULL;
      }
       else
      {
            u->index = v1->index;
            if (v1->index==0)
           u->index = v2->index;
            else
           if (v2->index==0)
                 u->index=v1->index;
           else
                 if (v1->index>v2->index)
                u->index=v2->index;
            if (u->index == v1->index)
           {
                 vlow1 = v1->low;
                 vhigh1 = v1->high;
           }
            else
           {
                 vlow1 = v1;
                 vhigh1 = v1;
           }
            if (u->index == v2->index)
           {
                 vlow2 = v2->low;
                 vhigh2 = v2->high;
           }
            else
           {
                 vlow2 = v2;
                 vhigh2 = v2;
           }
            u->low = Apply_Step(vlow1, vlow2, fct);
            u->high = Apply_Step(vhigh1, vhigh2, fct);
      }
       return(u);
}
```

c) Elimination d'une variable

L'élimination d'une variable dans une fonction produit une nouvelle fonction qui correspond à l'union de toutes les fonctions attachées à une valeur de vérité de cette variable.
Soit $f(x_1,...,x_n)$ la fonction, et x_i la variable:

$$f(x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = f_{|xi=true} \vee f_{|xi=false}$$

Il faut cependant arriver à déterminer la représentation d'une fonction (ou restreindre la fonction) $f(x_1,...,x_n)$ pour une valeur de vérité associé à une variable x_i (c'est-à-dire calculer $f_{|xi=true}$ ou bien calculer $f_{|xi=false}$). Grâce à la structure de nos arbres, ce calcul est très facile. En effet, chaque fois que l'on se retrouve avec un pointeur vers un sommet s d'index i, il suffit de "repointer" vers *low(s)* dans le cas de $f_{|xi=false}$, ou vers *high(s)* dans le cas de $f_{|xi=true}$.
Connaissant ces deux "sous-fonctions", il nous reste à appliquer l'algorithme du chapitre précédent avec la fonction OR pour obtenir notre fonction finale.

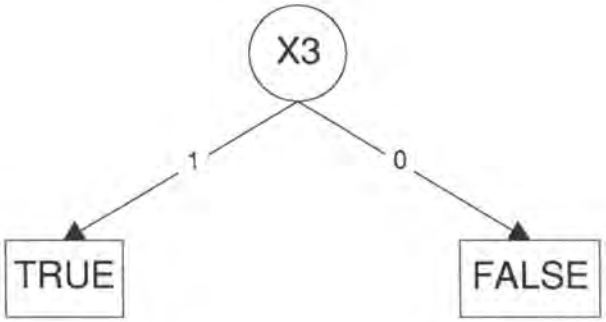
Hélas, ce n'est pas performant. Plutôt que de créer les deux sous-fonctions puis de les "OR-er", nous allons exécuter ces deux opérations en une seule phase.
Nous allons utiliser simplement une variante de la fonction APPLY, où les repointages vers *low* ou vers *high* sont intégrés et où la fonction OR est directement utilisée.

3. Utilisations

Pour faciliter l'utilisation de ces formules logiques, nous avons mis à disposition un ensemble de fonctions directement utilisables: un constructeur, un destructeur, les opérateurs OR, AND, EQU, NOT.

Une fonction est utilisée via un pointeur nommé `PFUNCTION` vers l'arbre.
Le constructeur produit un arbre correspondant à la fonction contituée de la seule variable dont l'indice est passé comme argument.

ex: si on désire la fonction $(X_3 \wedge TRUE) \vee (X_3 \wedge FALSE)$, on utilise `NewFunction(3)`



Les opérateurs fournissent une nouvelle fonction relative à l'opération désirée sur les deux fonctions passées comme paramètres, excepté pour NOT qui agit directement sur son argument.
Nous fournissons aussi une fonction qui affiche une fonction donnée sur un télétype: seul les opérateurs OR et AND et NOT sont représentés (OR par ".", AND par "+", et NOT par "|")

4. exemple

si on désire calculer $(x1 \wedge x2) \vee (\neg x3 \wedge x4)$

```
{       PFunction f1, f2, f3, f4, a1, a2, o1;
        f1 = NewFunction(1);
        f2 = NewFunction(2);
        f3 = NewFunction(3);
        f4 = NewFunction(4);
        a1 = AND(f1, f2);
        NOT(f3);
        a2 = AND(f3, f4);
        o1 = OR(a1,a2);

        printf("\n\n");
        AfficheFunction(stdout,o1);
        DeleteFunction(f1);
        DeleteFunction(f2);
        DeleteFunction(f3);
        DeleteFunction(f4);
        DeleteFunction(a1);
        DeleteFunction(a2);
        DeleteFunction(o1);
}
```

produira :

X1.(X2+!X2.(!X3.(X4)))+!X1.(!X3.(X4))

B. L'algorithme d'interprétation abstraite

Au Chapitre III.A.2.c, nous avions découvert un algorithme permettant de trouver les solutions pour la résolution d'un programme en fonction d'un but donné. C'est cet algorithme que nous avons voulu implémenter.

Au départ de ce travail, nous nous sommes fixé un certain nombre d'objectifs:

Le premier, le principal, était de rester aussi proche que possible de l'esprit de l'algorithme imaginé par le professeur Filé, et que nous avons tenté de vous présenter au début de cet ouvrage. Par proche, nous entendons à la fois, d'une part, le respect de la structure et du fonctionnement de l'algorithme, mais surtout, d'autre part, la conservation de la "généricité" et l'indépendance du choix du domaine de calcul à appliquer.

Le deuxième objectif trouve son origine dans l'existence d'autres algorithmes d'interprétation abstraite. Plus spécifiquement, des algorithmes utilisant une sémantique de point fixe* ont été développés et ont fait l'objet de nombreux travaux fructueux, dont un, en particulier, se trouve présenté dans [NELISS91]. Il nous semblait clair, ou du moins intéressant par certains aspects, de conserver une certaine homogénéité avec ces travaux. Non pas une compatibilité à toute épreuve, mais au moins un moyen éventuel de comparaison.

Finalement, nous avons désiré garder au programme une possibilité d'adaptation, d'évolution face aux besoins à venir.

La résolution du premier objectif imposait de garder séparées la gestion du domaine de calcul et la gestion du programme. Nous avons donc essayé de concevoir ce programme selon une méthodologie dite "orientée objet" ou "programmation logique"**. Ceci permettait d'avoir en tête l'idée d'existence de valeurs abstraites, de pouvoir les conserver, de les utiliser, et ce, sans pour autant en connaître l'implémentation ni même la sémantique intrinsèque.

Pour le deuxième objectif, il nous a semblé que conserver le même type d'input que ceux utilisés par Nélissen dans [NELISS91], ainsi que le langage utilisé pour la programmation est un bon procédé pour approcher l'homogénéité désirée.

Nous allons, maintenant, vous présenter, dans un premier temps la structure et la syntaxe d'un programme logique qui est accepté en entrée. Ensuite, dans un second temps, les structures des données utilisées dans le programme d'interprétation.

---

* La théorie de cette méthode d'interprétation, ne faisant pas partie du cadre du travail présent, n'a pas été présentée dans cette ouvrage. Cependant, le lecteur désireux de se faire une idée du fonctionnement et du raisonnement de se système, peuvent se référer aux articles [LEMUVA90], [LEMUVA91], [LECVAN92] .

** A ne pas confondre avec la programmation dans un "langage logique" comme PROLOG

1. Syntaxe d'un programme logique

Comme expliqué précédemment, nous désirions absolument conserver la structure de programme logique utilisée dans d'autres travaux. Dans [MUSUMB90], on retrouve la déclaration de programme logique normalisé. C'est une structure très claire, et qui permet une analyse syntaxique extrêmement facile. Avant de la détailler, nous allons succinctement la présenter. On peut sentir deux idées maîtresses, dans la différenciation qui existe entre programme logique normalisé et les programmes logiques "standard" (tel PROLOG) décrit par Lloyd dans [LLOYD87]:

- d'abord, toutes les variables ont comme nom X suivi d'un nombre,
- ensuite, toutes les substitutions doivent être explicites, ce qui signifie que l'entête ne contient que des variables, ainsi que les prédicats constituant les clauses.

a) Description:

Soit F_i est l'ensemble des fonctions d'arité i.($i \geq 0$)
Soit P_i est l'ensemble des symboles prédicatifs d'arité i.($i \geq 0$)
Soit VP est l'ensemble infini des variables de programme avec $x_1,...,x_n \in VP$
Soit VR est l'ensemble infini des variables de renommage avec $y,z... \in VR$

Un *programme logique normalisé* est un ensemble fini de procédures normalisées.
Une *procédure normalisée* est une suite non vide de clauses normalisées de même foncteur
Une *clause normalisée* a la forme :

$p(x_1,...,x_n) \leftarrow SAN$

où $n \geq 0$, $p \in P_n$ et SAN est une Suite d'Atomes Normalisés.
Si il existe m variables distinctes dans la clause normalisée, alors ces variables doivent être des variables de programme ($x_1,...,x_m \in VP$).

Un *atome normalisé* est soit un built-in, soit un appel de procédure.
Un *built-in normalisé* a la forme :

$x_i = x_j$ (i=j)
ou $x_i = f(x_{j1},...,x_{jn})$
où $x_i,x_{j1},...,x_{jn}$ sont des variables distinctes et $f \in F_n$.

Un *appel de procédure normalisé* a la forme :

$p(x_{i1},...,x_{in})$
où $x_{i1},...,x_{in}$ sont des variables distinctes et $p \in P_n$.

Un *terme normalisé* est soit une variable, soit une construction de la forme :

$f(t_1,...,t_n)$
où $f \in F_n$ et $t_1,...,t_n$ sont des termes normalisés.

Un *foncteur* est caractérisé par son nom et son arité, c'est-à-dire le nombre de ses arguments.
Une *constante* est un foncteur d'arité nulle.

b) Extension

Pour construire des listes, il existe le foncteur de construction de liste "."
Nous introduisons ici une représentation plus pratique, et équivalente à ce foncteur.
Une *liste* est
soit la *liste vide*, dénotée []
soit une structure à deux composants : sa tête et sa queue

On indique la fin d'une liste en lui donnant pour queue la liste vide.
Une liste non vide est représentée par le foncteur "." dont
le premier argument est la tête de la liste,
le second argument en est la queue.

Ex: ".(T,[])" correspond à la liste du seul élément T; que l'on peut aussi représenter par "[T]".
"[X1,X2,X3]" correspond à ".(X1,.(X2,.(X3,[])))" c'est-à-dire une liste de 3 éléments.

La notation "[T|Q]" une liste de tête T, et de queue Q.

Ex: "[1,2,3,4]= [T|Q]" représente l'unification de la variable T avec [1] comme tête de liste, et de la variable Q avec [2,3,4] comme queue de la liste.

c) Syntaxe

```
Zéro          = "0".
ChiffreNonNul = "1" |...| "9".
Chiffre       = Zéro | ChiffreNonNul.
Entier        = Zéro | (ChiffreNonNul { Chiffre}).
LettreMin     = "a" |...| "z".
AlphaNum      = LettreMin | Chiffre.
SymbVariable  = "X" {Entier}.
ListeVide     = "[" "]".
ListeUnitaire = "[" SymbVariable "]".
ListeMultiple ="[" SymbVariable
                 "|" SymbVariable {"," SymbVariable} "]"
Liste         = ListeVide | ListeUnitaire |    ListeMultiple
Constante     = Entier | ListeVide
SymbPred      = LettreMin { AlphaNum |"_"}
ListeVar      = "("SymbVariable { ","SymbVariable}")"
UnifVar       = SymbVariable "=" Symbvariable
UnifFonct     = SymbVariable "=" Liste
UnifConst     = SymbVariable "=" SymbPred
BuiltIn       = UnifVar | UnifFonct
AppelProc     = SymbPred ListVar
AtomeNorm     = BuiltIn | AppelProc
ClauseNorm    = AppelProc (":-") "(" {AtomeNorm} ")""."
```

ProcNorm = ClauseNorm {ClauseNorm}
PgmNorm = ProcNorm {Procnorm}

d) Exemple de programme normalisé

Voici un programme normalisé composé d'une procédure normalisée, elle-même divisée en deux clauses normalisées.

$append(x_1,x_2,x_3)\ \ x_1=[],\ x_2=x_3.$
$append(x_1,x_2,x_3) \leftarrow x_3=[x_4|x_6],append(x_5,x_2,x_6),x_1=[x_4|x_5]$

$F_3=\{[]\}$
$P_3=\{"append"\}$
$VP=\{x_1,x_2,x_3,x_4,x_5,x_6\}$
$VR=\{\}$

La première clause normalisée est formée de

un en-tête : $append(x_1,x_2,x_3)$ où "append" est un symbole prédicatif et x_1,x_2,x_3 sont des variables de programme (=expression de append/3).
un corps : formé lui-même de deux atomes normalisés
$x_1=[]$: est un built-in normalisé où x_1 est une variable de programme et [] représente une constante c'est-à-dire un foncteur 0-aire.
$x_2=x_3$: est un built-in normalisé où x_2 et x_3 sont des variables de programme

La deuxième clause est formée de

un en-tête : append/3
un corps : formé lui-même de trois atomes normalisés
$x_3=[x_4|x_6]$: est un built-in normalisé où $[x_4|x_6]$ est équivalent à ".(x4 x6)" avec "." un foncteur et $\{x_3,x_4,x_6\}$ des variables de programme.
append (x_5,x_2,x_6) : est un appel de procédure normalisé où append est un symbole prédicatif; $\{x_2,x_5,x_6\}$ sont des variables de programme.

$x_1=[x_4|x_5]$: est un built-in normalisé où $[x_4|x_5]$ est équivalent à ".(x4,x5)" avec "." un foncteur et $\{x_1,x_4,x_5\}$ des variables de programme.

2. Structures de données

Les structures de données se divisent en deux grandes classes:
les structures nécessaires à l'interprétation tabulée (Call-exit marker, arbre, table),
et les structures de représentation d'un programme logique(programme, clauses, but)

La seconde classe s'explique par la nécessité d'accéder de manière efficace au programme source.
Ce dernier sera donc, avant toute interprétation, "pre-compilé" en une "représentation machine" plus utile.

a) Programme

Le programme se compose d'un tableau reprenant toutes les procédures du programme source. Toutes les clauses de même nom et de même arité forment une procédure.

Une ligne de ce tableau comprend les éléments suivant:

1. le nombre de clause constituant la procédure,
2. l'arité de cette procédure,
3. un index vers la table des symboles (table où sont rassemblés tous les termes trouvés dans le programme source) pour connaître le nom de la procédure.
4. enfin une table constituée d'autant de lignes qu'il y a de clauses.

Une clause est définie comme suit: $H(Y) \leftarrow c, q_1(Y_1), ..., q_n(Y_n)$. L'en-tête est déjà défini, il ne reste plus qu'à représenter le corps de chaque clause.

Chaque ligne définissant une clause se compose de trois attributs:

1. le nombre total de variables utilisées dans cette clause,
2. la représentation de c,
3. la liste des "buts" $q_i(Y_i)$, dont la structure est définie au paragraphe suivant.

```
typedef struct tClause
{ int       nbvar;
  PAbstract beta;
  PLGoal    buts;
} Clause;

typedef struct tClause *PClause;
typedef struct tClause *PClauseList;


typedef struct tProc
{ int         nbclauses;
  int         nomclause;
  int         arite;
  PClauseList clauses;
} Proc;

typedef int   IProc;
typedef Proc *PProc;
typedef Proc *TProc;


typedef struct tPgm
{ int   nbproc;
  TProc procedure;
} Pgm;
```

b) But

Avant de commencer l'explication de la représentation d'un but, il convient de mettre chacun en garde contre l'abus (volontaire) qui a été fait du terme but dans ce chapitre. En effet, selon la définition donnée lors du chapitre III.A. paragraphe *programmes et calculs.*, un But est un élément de la forme $\leftarrow c,q_1(Y_1), ...,q_n(Y_n)$. L'ensemble des éléments $q_i(Y_i)$ forme le But. De même, la substitution du left-most atom de ce But avec une clause du programme fournit un nouveau But. Nous avons (très subjectivement d'ailleurs) décidé de nommer aussi cet élément $q_i(Y_i)$ un but (que nous différencierons de l'autre par l'usage d'un minuscule).

Dans un but, on peut trouver différents types d'éléments:

un appel à un clause du programme de la forme *terme(liste-de-variable)*
un built-in d'unification de variables de la forme *variable=variable*
un built-in d'unification avec une constante de la forme *variable=terme*
un built-in d'unification avec un foncteur de liste de variables de la forme
variable=[var$_1$, ..., var$_n$]
un built-in d'unification avec un autre foncteur de la forme
variable=terme(var$_1$, ..., var$_n$)

Nous avons utilisé une seule et même structure pour exprimer toutes ces variations; l'analyse du contenu de chacun des champs permet de fournir le type de but représenté.

```
typedef struct tGoal
{
    IProced  procedure;
    VarListe VL;
    Variable Var1;
} Goal;
```

Voici un tableau qui permet d'expliquer le fonctionnement de cette structure. La clef de lecture est la suivante: la première colonne contient les différents champs, la première ligne présente tous les types de buts (avec un exemple), enfin les intersections fournissent le contenu des champs dans le cas de figure correspondant.

type de but	*exemple*	*procedure*	*VL*	*Var1*
appel de clause	append(X1,X2,X3)	pointeur vers procedure append/3	X1, X2, X3	Nul
unif. de variable	X2=X3	Nul	X3	X2
unif. avec terme	X1= bonjour	pointeur vers symbole bonjour	Nul	X1
unif. liste variable	X3=[X5|X6]	Nul	X5, X6	X3
unif. built-in	X2=add(X1,X3)*	pointeur vers symbole add	X1, X3	Nul

Pour être exact, les pointeurs sont effectivement des index vers les lignes, respectivement, du tableau des procédures, du tableau des symboles, du tableau des built-in.

* En supposant bien entendu que add soit un built-in reconnu.

c) Call-Exit Marker

Le call-exit Marker est sans aucun doute la structure la plus facile à mettre en oeuvre.
Le call-exit Marker (ou CEXM) représente la substitution faite durant l'exécution du programme entre un but et une clause du programme. Ce CEXM est de la forme [A=H,c]• où H est l'en-tête de la clause, A le but d'appel de clause, et c une valeur du domaine de calcul.
ex : [append(X1,X2,X3) = append(X4,X5,X6), val]•.

Nous n'avons besoin que de quatre éléments: un pointeur vers la clause dont H n'est qu'une variante, la liste des variables de A, celles de H, et enfin un pointeur vers *val*.

```
typedef struct tExitMarker
{
    IndexProcedure   procedure;
    VarListe  V1;
    VarListe  V2;
    PAbstract val;
} ExitMarker;
```

Dans l'exemple précédent, *procedure* pointera (en pratique un index) vers append/3, *V1* contiendra [X1,X2,X3], *V2* contiendra [X4,X5,X6]. *val* s'explique de soi.

d) Arbre

Dans l'algorithme du chapitre III.A.2, l'ensemble des Buts $\leftarrow c,q_1(Y_1), ...,q_n(Y_n)$ à résoudre se retrouve au sein d'un arbre. Chaque But constitue un noeud de cet arbre, et pour chaque voie de résolution possible (entendez par là "pour chaque substitution possible") se trouve une nouvelle branche à l'arbre.

C'est une technique habituelle en programmation que d'implémenter un arbre sous la forme d'une liste chaînée. Le type de chaîne dépend du type de parcours à exécuter sur l'arbre* .
Dans le cas présent, hormis l'accès aux noeuds *référence*, le seul besoin est de créer des descendants sans retour aux ancêtres. Nous utiliserons donc une liste simplement chaînée comme implémentation.

Cependant, cette représentation nous fait perdre une information importante: la filiation. Il est certain que si l'on recherche seulement les solutions d'un But, nous n'avons nulle nécessité de nous souvenir de toutes les étapes nécessaires à la résolution, et la liste simple est suffisante. Pourtant, nous avons pensé que conserver uniquement les différents noeuds n'était pas suffisant. Nous avons donc ajouté des attributs à la structure suivante, représentant un noeud, de manière à conserver l'historique de la formation de notre arbre.

```
typedef struct tTreeNode
          {
             PLGoal     but;
             PAbstract  valeur;
             ITableLine reference;
             int        ident;
             PTreeNode  suivant;
             PTreeNode  son;
             PTreeNode  nextson;
```

* Une chaîne doublement chaînée si on doit pouvoir voyager dans tous les sens sur les branches de l'arbre. Une chaîne simplement chaînée suffit si le parcours est uniquement de type "top-down"

```
} TreeNode;
```

Finalement l'arbre se trouve en mémoire avec la structure suivante:

e) Table

Toujours dans l'algorithme décrit au chapitre III.A.2., on décrit une table où chaque ligne est constituée de deux éléments:
un call-pattern qui fait office de clef d'accès à la table,
et une liste de solutions associées à ce CEXM.

On accède à la liste de solutions dans deux grands cas: un noeud référence qui utilise ces solutions, et un noeud spécial qui ajoute éventuellement une solution. Le premier cas ne pose pas de problème; le second, par contre, nécessite de retrouver tous les noeuds références qui utilisent cette liste de solutions. Pour éviter de devoir parcourir tout l'arbre à la recherche des éventuels noeuds références, nous avons préféré ajouter une table de pointeurs vers tous ces noeuds. L'accès est ainsi grandement facilité et accéléré.

```
typedef struct tTableLine
          {
            PPattern   entree;
            TSolution  solutions;
            TRefTable  users;
          } TableLine;
```

f) Valeur Abstraite

En respect avec notre premier objectif, nous ne décrirons pas le domaine de calcul.
Pourtant nous utilisons bien un domaine, mais nous le faisons au travers d'un interface.

C'est cet interface que nous allons décrire.

Puisque lors de la compilation il nous est impossible de connaître la taille exacte d'un élément du domaine de calcul, il semble évident que chaque fois que nous avons besoin de conserver une valeur de ce domaine, c'est un simple pointeur (dont la taille est fixe) vers une de ces valeurs que nous utilisons. Chaque fois que l'algorithme d'interprétation communiquera avec le domaine, ce seront des pointeurs qui seront échangés avec les fonctions interfaces.

Comme toute chose, le domaine a sans doute besoin d'être initialisé. Ce qui est fait par l'intermédiaire de la fonction `Init_Abstract_Domain()`. Lorsque le domaine ne sera plus nécessaire, il pourra être éliminé par un appel à `Remove_Abstract_Domain()`.
Suivant la définition d'un système de calcul défini à la page 14, il existe deux fonctions:
- la projection, qui sert à exprimer une valeur du domaine en fonction d'un certain nombre de variables,
- et la composition, qui est un opérateur diadique.

Ces deux fonctions ne sont cependant pas suffisantes. En effets, dans l'algorithme, il est fait appel à des notions d'équivalences; de plus cet algorithme est prévu pour l'interprétation de programmes logiques, il faut donc pouvoir exprimer des unifications dans le domaine en question. Additionnellement, deux fonctions utilitaires sont nécessaires: une "détruisant" une valeur définie dans le domaine, l'autre permettant de créer une copie d'une valeur.

```
void        Init_Abstract_Domain();
void        Remove_Abstract_Domain();

BOOL        IsEquivalent(PAbstract, PAbstract);
void        DeleteAbstract(PAbstract);
PAbstract   AbstractCopy(PAbstract, int);
```

AbstractCopy, non seulement fournit une copie d'une valeur, mais en plus l'adapte. Sachant que nous représentons des variables par des numéros, on fournit aussi un nombre indiquant sur quelle variable nous désirons que notre variable soit adaptée.
exemple: nous avons la valeur val ={X1/X2, X4/bonjour} et nous désirons un copie adaptée à la variable X5, pour obtenir la valeur val2={X6/X7, X9/bonjour}. On utilise val2=AbstractCopy{val,5).

La composition est définie comme une fonction diadique. La définition paraît simple. Comme l'algorithme utilise des formules du genre $c \oplus c' \oplus (A=H)$, il faut donc aussi définir une fonction pour exprimer la correspondance entre une équivalence de liste de variables ($A_1=H_1$, ..., $A_n=H_n$) et une valeur du domaine de calcul.

```
PAbstract Composition2(PAbstract, PAbstract);
PAbstract DoEquivalence(VarListe,VarListe);
```

Lors de la mise en forme du programme, un désir a été émis de la part de "futurs" utilisateurs. Plutôt que de faire appel deux fois à la fonction de composition et une fois à la fonction DoEquivalence, pourquoi ne pas fournir tous les paramètres d'un coup, et pouvoir faire le calcul en bloc?
Si une personne met au point un domaine de calcul, nous lui laissons donc le choix de la méthode. Des switches (#define) sont prévus dans le programme implémentant l'algorithme. Il suffit de les changer pour utiliser l'un ou l'autre à la compilation. Il faut alors tenir compte de ce que certains paramètres peuvent être nuls, puisqu'il existe dans l'algorithme des formules telles que $c \oplus (A=H)$.

```
PAbstract Composition(PAbstract, PAbstract, VarListe,
VarListe);
```

Les unifications se font grâce aux fonctions suivantes** :

```
PAbstract UniVar(Variable, Variable);
PAbstract UniFonct(Variable, VarListe);
PAbstract UniConst(Variable, char*);
```

UniVar donne la représentation dans le domaine de calcul donné des formules du type $X_i=X_k$.
UniFonct s'utilise lorsqu'une variable s'unifie avec un foncteur, en particulier le foncteur de liste.
UniConst correspond à l'unification d'une variable avec un terme constant.

Dans le cadre de ce travail, nous avons décidé d'utiliser le domaine abstrait PROP. Pour le fonctionnement théorique de Prop, on peut se référer au chapitre II.2 à la page 8. En pratique, nous utilisons la représentations de formules logiques présentées au chapitre IV.A

** Ces fonctions ne sont pas nommée AIVAR ou AIFONCT, comme c'est le cas dans d'autre algorithme d'interprétation abstraite, pour souligner que le domaine de calcul n'est pas forcément un domaine qui en simule un autre.

Notre domaine ne nécessite pas d' Initialisation (et donc pas d'élimination non plus), puisque nous représentons chaque valeur abstraite par une fonction indépendante. Pour la projection, nous créons d'abord une liste des variables présente dans la valeur abstraite, ensuite nous éliminons une à une, celles qui ne sont pas présentes dans la liste sur laquelle a lieu la projection.
Pour le reste, nous utilisons directement les fonctions AND, OR, EQU offertes par l'implémentation des formules logiques.

3. L'algorithme

a) Fonctionnement

Avant toute chose, le programme logique est inséré en mémoire sous une forme digérée, dans les structures de programme décrites précédemment.
Ensuite, pour chaque clause, les substitutions sont "abstractisées". Il ne restera plus que les prédicats au sein des clauses.
Il en est de même avec le But à résoudre.

Puis l'algorithme décrit au chapitre 1.0.2.3. est utilisé: on crée d'abord un tableau vide et la racine de l'arbre, puis on étend l'arbre et la table.

Précédemment, nous avons dit que l'arbre est représenté sous forme d'une liste chaînée. La manière dont sont insérés les nouveaux noeuds dans cette chaîne cadre avec la méthode d'analyse dans l'arbre de recherche. En effets, le parcours d'un arbre peut être de type "en profondeur d'abord", ou bien de type "en largeur d'abord". Nous avons choisi cette dernière, car c'était de loin la plus facile à implémenter. Nous ajoutons tous les nouveaux noeuds en bout de liste, or pour la résolution d'un but donné, nous opérons toutes les opérations de substitutions en un coup (plutôt que d'abord résoudre, avant de faire une nouvelle substitution). Cette liste est parcourue séquentiellement. Le tout mis ensemble s'assimile à un parcours "en largeur d'abord".

b) Changement de Variables

Lorsqu'un atome s'unifie avec une clause, les variables {1...n} de cette clause sont renommées. Ce renommage doit se faire de telle manière qu'il n'y ait pas de conflits avec des variables déjà utilisées lors de l'éxécution. Puisque nous n'utilisons que des variables constituées d'un numéro, il nous suffit de n'assigner un "nom" aux variables que de manière incrémentale, au fur et à mesure, suivant l'ordre temporel d'apparition.La seule information nécessaire est le numéro de la dernière variable utilisée. Les "noms" des nouvelles variables, pour être différents, devront seulement avoir comme chiffre, un nombre plus grand que celui de la dernière variable utilisée.

On retrouve la table de renommage, conséquence d'une substitution, dans les call-exit marker.

C. Utilisation

1. Résultats fournis

Il y a deux types de résultats: la table et l'arbre générés par l'algorithme.

a) La Table

La table se compose de lignes. Chaque ligne elle-même, est composée de deux éléments:un call-pattern et une liste de solutions.

Les lignes sont affichées sous le format suivant:

```
==============      Ligne N  =============
terme(liste-de-variables)
substitution abstraite d'entrée
-----  k solutions
1ere solution
     .
     .
kieme solution
```

avec en italique les zones variables.

ex: soit la ligne suivante, 3ieme de la table, où deux solutions sont associées au call-pattern

[exemple(var1,var2), var1/constante]•	var1/constante var1/var2

L'affichage sera

```
                .
                .
==============      Ligne 3  =============
exemple(var1,var2)
var1/constante
-----  2 solutions
var1/constante
var1/var2
                .
                .
```

2. L'Arbre

Il y a deux informations nécessaires dans un arbre: le contenu de chaque noeud et la structure de l'arbre.
Pour afficher l'arbre, nous avons désiré afficher la struture où chaque noeud est représenté par un identifiant, ensuite afficher linéairement la longue suite du contenu des noeuds (toujours avec leur identifiant pour pouvoir les reconnaître)

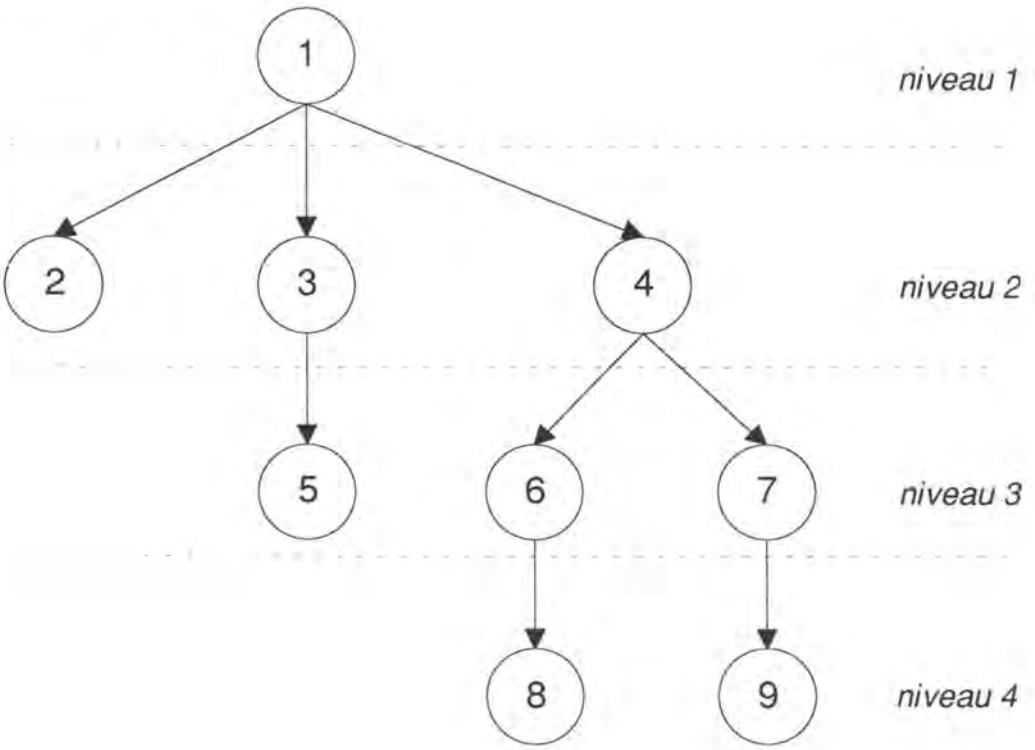
a) Structure

Un arbre est une racine reliée à des sous-arbres.

On affiche la racine, puis on affiche toute la structure du premier sous-arbre, suivie de l'affichage de la structure de deuxième, et ainsi de suite jusqu'au dernier sous-arbre.

Tous les noeuds de même niveau, sont alignés sur une même colonne. Par niveau, nous entendons que si le noeud-racine de l'arbre est de niveau *i*, alors les noeuds-racines des sous-arbres sont de niveau *i+1*.

ex: si nous devons afficher l'arbre suivant



nous obtiendrons ceci:

```
-->1-->2
   +-->3-->5
   +-->4-->6-->8
       +-->7-->9
```

b) Description d'un noeud

Un noeud est libellé par un But généralisé, soit de la forme $\leftarrow c, A_1, ..., A_k$ (voir chapitre concernant l'interpréteur tabulé). A l'affichage, nous distinguons bien la valeur du domaine de calcul *c* de la liste des A_i. A_i est soit un atome, soit une call-exit marker.

ex: soit le But suivant, étiquettant le noeud numéroté 3

← {X3/piston, X4/cylindre, X1/oldsmobile},
madeof(X3,X4), [haveamotor(X1)=haveamotor(X2), {X1/oldsmobile}]•, canride(X1).

sera affiché

```
                    .
                    .
**********    Noeud 3    **********

valeur: {X3/piston, X4/cylindre, X1/oldsmobile}
liste goals:
        madeof(X3,X4)
        [haveamotor(X1)=haveamotor(X2),{X1/oldsmobile}]
        canride(X1)
                    .
                    .
```

D. Mise en oeuvre

Tout d'abord il faut choisir le programme logique que l'on désire interpréter. Ainsi que le But avec lequel on veut l'interpréter.

Le programme doit se trouver dans un fichier de texte, et sa syntaxe doit être celle des programmes logiques normalisés à une variante près. Une ajoute simple doit être insérée dans la syntaxe: avant chaque clause, il faut ajouter un bloc de la forme "{x,y}" où *x* représente le nombre de variables présentes dans l'en-tête de la clause (autrement dit l'arité de la clause), et *y* le nombre de variables présentes dans le corps de la clause et qui ne se trouvent pas dans l'en-tête.

ex: le programme append, dont la version normalisée se trouve page 14

devra être remanié

```
{3,0}
append(X1,X2,X3):-X1=[],X2=X3.
{3,3}
append(X1,X2,X3):-X1=[X4|X5],X3=[X4|X6],append(X5,X2,X6).
```

Cette règle est aussi valable pour le But à résoudre, à la différence que le bloc est de la forme {x}, où x est le nombre de variables présentes dans le But.

ex:

```
{3}
:- X1=dummystring, X2=X1, append(X1,X2,X3).
```

1. Emploi

L'introduction du But peut se faire de deux manières: soit elle se trouve dans un fichier texte, soit on l'introduit au clavier lors de la résolution. Par défaut, l'entrée se fait au clavier.

L'affichage des résultats (table et arbre) peut se faire soit à l'écran soit dans des fichiers textes. Par défaut, l'affichage se fait à l'écran.

Voici le synopsis de la ligne de commande:

TABUL *nom_fichier_programme* [-T*nom_fic_table*] [-A*nom_fic_arbre*] [-g*nom_fic_but*]

E. Modification de domaine abstrait

Lors de l'implémentation, nous avons désiré utiliser l'interpréteur avec le domaine abstrait PROP.

Il est possible, si on le désire, d'utiliser l'algorithme avec un autre domaine de calcul. Il faut, alors, créer le programme de gestion du domaine que l'on souhaite appliquer.
Ensuite il faut écrire les procédures qui font l'interface entre l'interpréteur et le domaine. La liste des fonctions à été donnée au chapitre IV.B.2.f.
Il faut aussi avoir un pointeur vers la structure qui représente une substitution abstraite. Ce pointeur est nommé `PAbstract`, car c'est le nom utilisé dans tous le code de l'interpréteur.

Dans PROP, par exemple, toutes la gestion consiste dans l'utilisation de formules propositionnelles. Et une substitution (ou valeur du domaine de calcul), en est simplement une formule.

F. Exemples d'utilisation

Nous allons à présent exposer quelques exemples d'utilisation de l'interpréteur avec le domaine abstrait PROP.

Nous envisagerons trois programmes: append, queens, quicksort. Le choix de ceux-ci n'est pas innocent. En effet, ils sont d'une difficulté croissante. Et leur analyse permettra d'apporter une indication* de la complexité, au sens habituel en informatique, d'une interprétation tabulée.

Pour chaque programme, nous envisagerons plusieurs Buts à résoudre.

Sauf pour le programme append, nous n'avons pas présenté ici les tables et arbres résultant de l'interprétation** . Nous nous sommes bornés à rassembler, les solutions finales de nos Buts à résoudre. Les résultats seront fournis sous forme d'une table où chaque ligne représente une interprétation. La première colonne correspond à la substitution d'entrée, la deuxième à la substitution résultat. La troisième colonne sera l'indicateur de complexité, constitué de deux nombres: le nombre de noeuds dans l'arbre, et le temps pour l'interprétation. Attention toutefois, ces temps sont ceux d'une interprétation exécutée sur une machine de type PC-386-33Mhz; de plus ces temps comprennent les actions d'entrée sortie.

---

* Nous ne prétendons pas apporter, ici, une affirmation ni une justification de cette complexité. Notre but est, simplement, de présenter ce qui nous apparaît être de type exponentiel.

** Nous les avons cependant mises en annexe.

G. APPEND

Nous allons nous concenter sur un But assez trivial: un simple appel à append. Nous varierons cependant la contrainte associée.

Commençons, en supposant la substitution d'entrée, telle que nous ne connaissons pas d'information sur la troisième variable: {X1/ground, X2/ground, X3/any}
Nous introduisons donc le But suivant:

```
{3}
:- X1=[], X2=[], append(X1,X2,X3)
```

rem: dans PROP, le terme associé à une variable n'a pas d'importance. Qu'on utilise une constante ou une liste vide ne change rien. Ceci n'est pas forcément vrai avec un autre domaine.

1. Exécution

a) Arbre

```
#################################################################
##                    STRUCTURE DE L'ARBRE                     ##
#################################################################

-->0-->2-->6-->7
       +-->4-->5
   +-->1-->3


#################################################################
##                  DECOMPOSITION DE L'ARBRE                   ##
#################################################################

**********      Noeud 0      **********

valeur:  X1.X2
liste goals:
         append(X1,X2,X3)
**********      Noeud 1      **********

valeur:  X4.X5.X6
liste goals:
         [append(X1,X2,X3) = append(X4,X5,X6), X1.X2 ]
**********      Noeud 2      **********

valeur:  X7.X8.[(X9.X10.X11.X12)+(|X9.X10.X11.|X12)]
liste goals:
         append(X11,X8,X12)
         [append(X1,X2,X3) = append(X7,X8,X9), X1.X2 ]
**********      Noeud 3      **********

valeur:  X1.X2.X3
liste goals:
**********      Noeud 4      **********

valeur:  X8.X11.X12
liste goals:
         [append(X1,X2,X3) = append(X7,X8,X9), X1.X2 ]
**********      Noeud 5      **********

valeur:  X1.X2
liste goals:
**********      Noeud 6      **********

valeur:  X8.X11
liste goals:
         [append(X1,X2,X3) = append(X7,X8,X9), X1.X2 ]
**********      Noeud 7      **********

valeur:  X1.X2
liste goals:
```

b) Table

```
==============   Ligne 1    ==============
append(X1,X2,X3)
X1.X2
----- 2 solutions
X1.X2.X3
X1.X2
```

Substitution d'entrée	*Substitution de sortie*	*Nombre de noeuds dans l'arbre / temps d'exécution*
X1	X1	7 / 0.88"
X2	X2	7 / 0.91"
X3	X2	7 / 0.89"
X1.X2	X1.X2	7 / 0.84"
X1.X2.X3	X1.X2.X3	5 / 0.75"
TRUE	TRUE	7 / 0.83"

H. QUEENS

```
{2,0}
queens (X1,X2) :-
      perm (X1, X2), safe(X2).

{2,0}
perm(X1,X2):-X1=[],X2=[].
{2,6}
perm(X1,X2):-X1=[X3|X4],X2=[X5|X6],X7=[X3|X4],
             delete(X5,X7,X8),
             perm(X8,X6).

{3,0}
delete(X1,X2,X3):-X2=[X1|X3].
{3,3}
delete(X1,X2,X3):-X2=[X4|X5], X3=[X4|X6], delete(X1,X5,X6).

{1,0}
safe(X1):-X1=[].
{1,3}
safe(X1):-X1=[X2|X3],X4=[] ,noattack  (X2,X3, X4), safe(X3).

{3,0}
noattack(X1,X2,X3):-X2=[].
{3,7}
noattack(X1,X2,X3):-X2=[X4|X5],
                    inegal(X1,X4),plus(X6,X4,X3),
                    inegal(X1,X6),plus(X7,X1,X3),
                    inegal(X4,X7),X10=[],plus(X 9,X3,X10),
                    inegal(X8,X9),
                    noattack(X1,X5,X8).

{3,0}
plus(X1,X2,X3):- X1=[], X2=[], X3=[].

{2,0}
inegal(X1,X2):- X1=[], X2=[].
```

Substitution d'entrée	*Substitution de sortie*	*Nombre de noeuds dans l'arbre*
X1	X1.X2	42 / 4.69"

I. QUICKSORT

```
{2,1}
quicksort(X1,X2) :- X3=[], quicksort(X1,X2,X3).

{3,0}
quicksort(X1,X2,X3) :- X1=[], X3=X2.
{3,6}
quicksort(X1,X2,X3) :- X1=[X4|X5],
                       partition(X5,X4,X6,X7),
                  quicksort(X7,X8,X3),
                       X9=[X4|X8],
                  quicksort(X6,X2,X9).

{4,0}
partition(X1,X2,X3,X4) :- X1=[], X3=[], X4=[].
{4,3}
partition(X1,X2,X3,X4) :- X1=[X5|X6],
                          X3=[X5|X7],
                          inegal(X5,X2),
                          partition(X6,X2,X7,X4).
{4,3}
partition(X1,X2,X3,X4) :- X1=[X5|X6], X4=[X5|X7],
                          inegal(X5,X2),
                          partition(X6,X2,X3,X7).

{2,0}
inegal(X1,X2) :- X1=[], X2=[].
```

Substitution d'entrée	*Substitution de sortie*	*Nombre de noeuds dans l'arbre*
X1	X1	654 / 1' 45.11"
X1.X2	X1.X2	1168 / 3' 36.97"
TRUE	TRUE	741 / 1' 48.24"

V. Conclusion

Résultats :

Nous avons vu que l'utilisation de l'interprétation abstraite avait pour but de faciliter la recherche de **propriétés** spécifiques d'un programme logique.
Dans ce mémoire, nous avons utilisé l'interprétation abstraite de façon particulière - à l'aide du domaine PROP - pour obtenir ces propriétés, et les résultats que nous avons obtenus sont intéressants
La principale remarque que nous pouvons formuler est que la présentation de ces résultats (c'est-à-dire les propriétés des programmes) n'est actuellement pas très pratique : suivant le programme donné, les résultats fournis sont plus ou moins complexes, il suffit pour s'en rendre compte de se référer à la trace provenant de l'interprétation du programme "Quick". Le but initial de l'interprétation abstraite n'est donc que partiellement atteint.

Une première solution consisterait, à notre avis, à créer un programme qui analyserait les traces pour en retirer la "substantifique moëlle". Une autre solution serait de modifier le programme d'interprétation pour qu'il donne immédiatement les résultats désirés. Mais ce n'était pas le but de notre travail.

Avenir technique du programme :

Ce programme a un mérite certain : il fonctionne, et nous avons donc atteint l'objectif de départ.

Notre but était d'implémenter l'algorithme du Professeur Filé et, surtout, de conserver le "mapping" entre l'implémentation et la théorie. Ce qui est réalisé.

Pourtant, si ce programme devait être utilisé plus intensivement, il conviendrait de l'améliorer encore afin de résoudre certains problèmes. La gestion de la mémoire, par exemple, n'est pas optimale, plusieurs structures faisant double emploi.

Avenir théorique du programme :

Notre programme simule des programmes logiques, mais ceux-ci contiennent une sémantique limitée par rapport à Prolog. Des travaux ont analysé la possibilité d'y ajouter le built-in "cut". Il aurait été intéressant de pouvoir travailler (aussi) à leur lumière. Le temps à manqué pour pouvoir tenir compte de cette nouvelle orientation.

On peut, enfin et surtout, envisager l'utilisation du programme avec d'autre domaines abstraits que PROP.

Eventuellement, on peut aussi le comparer avec d'autres programmes utilisant d'autres méthodes d'interprétations.

Nous espérons que l'ouverture que nous avons laissée pour l'adaptation du programme pourra aider à son perfectionnement.

Pour ma part, il me semble que cette nouvelle voie ouverte est très prometteuse, et ouvre le champs à beaucoup de progrès...

BIBLIOGRAPHIE

VI. Bibliographie

[BRYANT86] : R. E. Bryant, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on Computers, Vol. C-35, N. 8, août 1986

[COFILE91] : A. Codognet, G. Filé, *Computations, Abstractions and Constraints in Logic Programs*, rapport interne 13, 1991

[COFOWI91] : A. Cortesi, G. Filé, W. Winsborough, *Prop revisited: Propositional Formula as Abstract Domain for Groundness Analysis*, in procs LICS 91,Sith annual IEEE symposium on Logic in Computer Science, Amsterdam, 1991

[CORTES92] : A. Cortesi, *Domini Astratti per I Analisi Statica di Programmi Logici*, thése de doctorat, Universita degli studi di Padova, Padoue, 1992

[LEMUVA90] : B. Le Charlier, K. Musumbu, P. Van Hentenryck, *Efficient and Accurate Algorithms for the Abstract Interpretation of Prolog Programs*, FUNDP, Namur, 1990

[LEMUVA91] : B. Le Charlier, K. Musumbu, P. Van Hentenryck, *A Generic Abstract Interpretation Algorithm and its complexity analysis,* Proceeding of the eighth International Conference on Logic Programming, ICLP 91, Paris, 1991

[LECHAR92] : B. Le Charlier, *L'analyse statique des programmes par interprétation abstraite*, FUNDP, Namur, 1992

[LECVAN92] : B. Le Charlier, P. Van Hentenryck, *Groundness Analysis for Prolog: Implementationand Evaluation of the Domain Prop*, FUNDP, Namur, 1992

[LECVAN92] : B. Le Charlier, P. Van Hentenryck, *On the Design of Generic Abstract Interpretation Frameworks*, in W.S.A. , Bordeaux, 1992

[LLOYD87] : J. W. Lloyd, *Foundation of Logic Programming*, seconde édition, Springer-Verlag, Berlin, 1987

[MUSUMB90] : K. Musumbu, *Interprétation Abstraite de Programme Prolog*, thèse de doctorat, FUNDP, Namur, 1990

[NELIS92] : P. Nélis, *Etude et Comparaison de deux Modèles d'Interprétation Abstraite*, mémoire, FUNDP, Namur, 1992

[NELISS91] : J-P. Nelissen, *Interprétation Abstraite: Analyse de Mode au moyende Formules Logiques*, mémoire, FUNDP, Namur, 1991

[STESHA86] : L. Sterling, E. Shapiro, *The Art of Prolog : Advanced Programming Techniques*, MIT Press, Cambridge, Ma., 1986

[LECH291] : B. Le Charlier, P. Van Hentenryck, *Experimental Evaluation of a Generic Abstract Interpretation Algorithm for Prolog*, FUNDP, Namur, 1991

ANNEXE

Trace de "append"

```
avec le but ":- X1=[], append(X1,X2,X3)."

################################################################
##                  STRUCTURE DE L'ARBRE                      ##
################################################################
-->0-->2-->6-->7
       +-->4-->5
   +-->1-->3
################################################################
##                DECOMPOSITION DE L'ARBRE                    ##
################################################################

**********     Noeud 0 **********

valeur: X1
liste goals:
                append(X1,X2,X3,•)
**********     Noeud 1 **********

valeur: X4.[(X5.X6)+(|X5.|X6)]
liste goals:
                [append(X1,X2,X3,•) = append(X4,X5,X6,•), X1 ]
**********     Noeud 2 **********

valeur: X7.[(X9.X10.X11.X12)+(|X9.X10.X11.|X12)]
liste goals:
                append(X11,X8,X12,•)
                [append(X1,X2,X3,•) = append(X7,X8,X9,•), X1 ]
**********     Noeud 3 **********

valeur: X1.[(X2.X3)+(|X2.|X3)]
liste goals:
**********     Noeud 4 **********

valeur: [(X8.X11.X12)+(|X8.X11.|X12)]
liste goals:
                [append(X1,X2,X3,•) = append(X7,X8,X9,•), X1 ]
**********     Noeud 5 **********

valeur: X1
liste goals:
**********     Noeud 6 **********
valeur: X11
liste goals:
                [append(X1,X2,X3,•) = append(X7,X8,X9,•), X1 ]
**********     Noeud 7 **********

valeur: X1
liste goals:

===============          Ligne 1          ==============
append(X1,X2,X3,•)
X1
----- 2 solutions
X1.[(X2.X3)+(|X2.|X3)]
X1
```

Trace de "append"

```
avec le but ":- append(X1,X2,X3)."

################################################################
##                  STRUCTURE DE L'ARBRE                      ##
################################################################
-->0-->2-->6-->7
       +-->4-->5
   +-->1-->3
################################################################
##                DECOMPOSITION DE L'ARBRE                    ##
################################################################

**********     Noeud 0 **********
valeur: --
liste goals:
               append(X1,X2,X3,•)
**********     Noeud 1 **********
valeur: X4.[(X5.X6)+(|X5.|X6)]
liste goals:
               [append(X1,X2,X3,•) = append(X4,X5,X6,•),  --  ]
**********     Noeud 2 **********

valeur:
       [(X7.[(X9.X10.X11.X12)+(|X9.X10.X11.|X12)])+(|X7.[(X9.X10.|X11.X12)+(|X9.[(X10.|X11.|X1
2)+(|X10)])])]
liste goals:
               append(X11,X8,X12,•)
               [append(X1,X2,X3,•) = append(X7,X8,X9,•),  --  ]
**********     Noeud 3 **********

valeur: X1.[(X2.X3)+(|X2.|X3)]
liste goals:
**********     Noeud 4 **********

valeur: [(X8.X11.X12)+(|X8.X11.|X12)]
liste goals:
               [append(X1,X2,X3,•) = append(X7,X8,X9,•),  --  ]
**********     Noeud 5 **********

valeur: 1
liste goals:
**********     Noeud 6 **********

valeur: 1
liste goals:
               [append(X1,X2,X3,•) = append(X7,X8,X9,•),  --  ]
**********     Noeud 7 **********

valeur: 1
liste goals:

===============         Ligne 1       ==============
append(X1,X2,X3,•)
 --
----- 2 solutions
X1.[(X2.X3)+(|X2.|X3)]
1
```

Trace de "queens"

```
avec le but ":- X1=[], queens(X1,X2).

##############################################################
##                    STRUCTURE DE L'ARBRE                  ##
##############################################################

-->0-->1-->3-->6->10->15->20->24->28->31
          +-->5-->9->14->19->23->27
      +-->2-->4-->8->13->18->22->26->30->32->33->34->35->36->37->38->39->40->41->42
                  +->12->17->21->25->29
             +-->7->11->16


##############################################################
##                 DECOMPOSITION DE L'ARBRE                 ##
##############################################################

**********      Noeud 0 **********

valeur: X1
liste goals:
                queens(X1,X2,•)
**********      Noeud 1 **********

valeur: X3
liste goals:
                perm(X3,X4,•)
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 2 **********

valeur: X5.X6
liste goals:
                [perm(X3,X4,•) = perm(X5,X6,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 3 **********

valeur: X7.[(X8.X9.X10.X11.X12.X13)+(|X8.X9.X10.[(X11.|X12.X13)+(|X11.X13)])]
liste goals:
                delete(X11,X13,X14,•)
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 4 **********

valeur: X3.X4
liste goals:
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 5 **********

valeur: X15.X16.X17
liste goals:
                [delete(X11,X13,X14,•) = delete(X15,X16,X17,•),
X7.[(X8.X9.X10.X11.X12.X13)+(|X8.X9.X10.[(X11.|X12.X13)+(|X11.X13)])] ]
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 6 **********

valeur: X19.[(X20.X21.X22.X23)+(|X20.X21.X22.|X23)]
liste goals:
                delete(X18,X22,X23,•)
                [delete(X11,X13,X14,•) = delete(X18,X19,X20,•),
X7.[(X8.X9.X10.X11.X12.X13)+(|X8.X9.X10.[(X11.|X12.X13)+(|X11.X13)])] ]
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 7 **********

valeur: X24
liste goals:
                [safe(X4,•) = safe(X24,•), X3.X4 ]
```

```
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 8 **********

valeur: X25.X26.X27.X28
liste goals:
                noattack(X26,X27,X28,•)
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 9 **********

valeur: X11.X13.X14
liste goals:
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 10        **********

valeur: X14.X18.X22.X23
liste goals:
                [delete(X11,X13,X14,•) = delete(X18,X19,X20,•),
X7.[(X8.X9.X10.X11.X12.X13)+(|X8.X9.X10.[(X11.|X12.X13)+(|X11.X13)])] ]
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 11        **********

valeur: X4
liste goals:
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 12        **********

valeur: X29.X30.X31
liste goals:
                [noattack(X26,X27,X28,•) = noattack(X29,X30,X31,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 13        **********

valeur: X32.X33.X34.X35.X36.X41
liste goals:
                inegal(X32,X35,•)
                plus(X37,X35,X34,•)
                inegal(X32,X37,•)
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 14        **********

valeur: X4.X12.X14
liste goals:
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 15        **********

valeur: X11.X13.X14
liste goals:
                perm(X14,X12,•)
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 16        **********

valeur: X1.X2
liste goals:
**********      Noeud 17        **********

valeur: X26.X27.X28
liste goals:
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
```

```
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 18        **********

valeur: X42.X43
liste goals:
                [inegal(X32,X35,•) = inegal(X42,X43,•), X32.X33.X34.X35.X36.X41 ]
                plus(X37,X35,X34,•)
                inegal(X32,X37,•)
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 19        **********

valeur: X3.X4
liste goals:
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 20        **********

valeur: X4.X12.X14
liste goals:
                [perm(X3,X4,•) = perm(X7,X8,•), X3 ]
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 21        **********

valeur: X27
liste goals:
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 22        **********

valeur: X32.X34.X35.X36.X41
liste goals:
                plus(X37,X35,X34,•)
                inegal(X32,X37,•)
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 23        **********

valeur: X4
liste goals:
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 24        **********

valeur: X3.X4
liste goals:
                safe(X4,•)
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 25        **********

valeur: X4
liste goals:
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 26        **********

valeur: X44.X45.X46
liste goals:
                [plus(X37,X35,X34,•) = plus(X44,X45,X46,•), X32.X34.X35.X36.X41 ]
                inegal(X32,X37,•)
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
```

```
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 27        **********

valeur: X1.X2
liste goals:
**********      Noeud 28        **********

valeur: X4
liste goals:
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 29        **********

valeur: X1.X2
liste goals:
**********      Noeud 30        **********

valeur: X32.X34.X35.X36.X37.X41
liste goals:
                inegal(X32,X37,•)
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 31        **********

valeur: X1.X2
liste goals:
**********      Noeud 32        **********

valeur: X32.X34.X35.X36.X37.X41
liste goals:
                plus(X38,X32,X34,•)
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 33        **********

valeur: X32.X34.X35.X36.X38.X41
liste goals:
                inegal(X35,X38,•)
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 34        **********

valeur: X32.X34.X35.X36.X38.X41
liste goals:
                plus(X40,X34,X41,•)
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 35        **********

valeur: X32.X34.X36.X40.X41
liste goals:
                inegal(X39,X40,•)
                noattack(X32,X36,X39,•)
                [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
                safe(X27,•)
                [safe(X4,•) = safe(X25,•), X3.X4 ]
                [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********      Noeud 36        **********

valeur: X47.X48
```

```
liste goals:
               [inegal(X39,X40,•) = inegal(X47,X48,•), X32.X34.X36.X40.X41 ]
               noattack(X32,X36,X39,•)
               [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
               safe(X27,•)
               [safe(X4,•) = safe(X25,•), X3.X4 ]
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 37       **********

valeur: X32.X36.X39.X40
liste goals:
               noattack(X32,X36,X39,•)
               [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
               safe(X27,•)
               [safe(X4,•) = safe(X25,•), X3.X4 ]
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 38       **********

valeur: X27.X32.X36.X39
liste goals:
               [noattack(X26,X27,X28,•) = noattack(X32,X33,X34,•), X25.X26.X27.X28 ]
               safe(X27,•)
               [safe(X4,•) = safe(X25,•), X3.X4 ]
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 39       **********

valeur: X26.X27.X28
liste goals:
               safe(X27,•)
               [safe(X4,•) = safe(X25,•), X3.X4 ]
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 40       **********

valeur: X27
liste goals:
               [safe(X4,•) = safe(X25,•), X3.X4 ]
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 41       **********

valeur: X4
liste goals:
               [queens(X1,X2,•) = queens(X3,X4,•), X1 ]
**********     Noeud 42       **********

valeur: X1.X2
liste goals:


===============         Ligne 1       ==============
queens(X1,X2,•)
X1
----- 1 solutions
X1.X2
===============         Ligne 2       ==============
perm(X3,X4,•)
X3
----- 1 solutions
X3.X4
===============         Ligne 3       ==============
delete(X11,X13,X14,•)
X13
----- 1 solutions
X11.X13.X14
===============         Ligne 4       ==============
safe(X4,•)
X4
----- 1 solutions
X4
===============         Ligne 5       ==============
noattack(X26,X27,X28,•)
X26.X27.X28
----- 1 solutions
X26.X27.X28
===============         Ligne 6       ==============
inegal(X32,X35,•)
X32.X35
----- 1 solutions
X32.X34.X35.X36.X41
===============         Ligne 7       ==============
plus(X37,X35,X34,•)
```

```
X34.X35
----- 1 solutions
X32.X34.X35.X36.X37.X41
==============        Ligne 8       ==============
inegal(X39,X40,•)
X40
----- 1 solutions
X32.X36.X39.X40
```

Trace de "quick"

avec le but ":- X1=[], quicksort(X1,X2)."

Cette trace est extrèmement longue. Nous l'avons insérée pour pouvoir comparer la compléxité avec les programmes plus simple comme "append"

```
#############################################################
##                  STRUCTURE DE L'ARBRE                   ##
#############################################################

-->0-->1-->3-->7->13->19->36->92>131>196>279>381>486>582
                                  +>278>380>485>581
                                     +>379>484>580
                            +->68>100>141>216>350>454>550
                                  +>215>303>405>508
                                     +>302>404>507
                            +->63->90>129>194>348>452>548
                                  +>193>275>375>480
                                     +>274>374>479
                            +->49->66->98>179>344>448>544
                                  +>139>212>299>401
                                     +>211>298>400
                      +->18->26->35->91>130>195>277>378>483>579
                                  +>276>377>482>578
                                     +>376>481>577
                         +->67->99>140>214>349>453>549
                                  +>213>301>403>506
                                     +>300>402>505
                         +->62->89>128>192>347>451>547
                                  +>191>273>373>478
                                     +>272>372>477
                         +->48->61->86>178>343>447>543
                                  +>124>184>265>361
                                     +>183>264>360
                      +->17->25->33->46->97>138>210>297>399>504>594
                                  +>209>296>398>503>593
                                +>137>208>295>397>502>592
                                  +>207>294>396>501>591
                             +->78>117>173>261>356>460>556
                                  +>172>260>355>459>555
                                +>116>171>259>354>458>554
                                  +>170>258>353>457>553
                             +->76>157>241>326>430>532>606
                                  +>240>325>429>531>605
                                +>113>165>253>352>456>552
                                  +>164>252>351>455>551
                             +->59>155>237>322>426>528>604
                                  +>236>321>425>527>603
                                +->83>121>180>345>449>545
                                  +>120>176>263>359>466>568>618>642>654
                                                    +>567>617>641>653
                                                 +>465>566>616>640>652
                                                    +>565>615>639>651
                                              +>358>464>564>614>638>650
                                                    +>563>613>637>649
                                                 +>463>562>612>636>648
                                                    +>561>611>635>647
                                              +>357>462>560>610>634>646
                                                    +>559>609>633>645
                                                 +>461>558>608>632>644
                                                    +>557>607>631>643
                                        +>262>341>445>541
                         +->45->58->82>119>175
                            +->81>118>174
      +-->6->10->12->87>125>186>346>450>546
                         +>185>267>363>468
                            +>266>362>467
                   +->34->47>177>342>446>542
                         +->60->85>123>182
                            +->84>122>181
                   +->16->24->32>106>149>228
                            +->44->57->80
                         +->31>105>148>227
                            +->43->56->79
      +-->5-->9->28>104>147>226
                   +->38->50->71
               +->11->15->23->42->94>133>200>287>389>494>590
                                        +>286>388>493>589
```

```
                                        +>199>285>387>492>588
                                           +>284>386>491>587
                                     +->70>102>145>224>311>413>516
                                           +>223>310>412>515
                                        +>144>222>309>411>514
                                           +>221>308>410>513
                                     +->65->96>136>206>293>395>500
                                           +>205>292>394>499
                                        +>135>204>291>393>498
                                           +>203>290>392>497
                                     +->55->77>115>169>257>340>444
                                           +>168>256>339>443
                                        +>114>167>255>338>442
                                           +>166>254>337>441
                               +->22->30->41->93>132>198>283>385>490>586
                                              +>282>384>489>585
                                           +>197>281>383>488>584
                                              +>280>382>487>583
                                  +->69>101>143>220>307>409>512
                                              +>219>306>408>511
                                           +>142>218>305>407>510
                                              +>217>304>406>509
                                  +->64->95>158>243>328>432>534
                                              +>242>327>431>533
                                           +>134>202>289>391>496
                                              +>201>288>390>495
                                  +->54->75>156>239>324>428>530
                                              +>238>323>427>529
                                           +>112>163>251>336>440
                                              +>162>250>335>439
                               +->21->29->40->88>127>190>271>371>476>576>626
                                                    +>370>475>575>625
                                           +>189>270>369>474>574>624
                                                    +>368>473>573>623
                                        +>126>188>269>367>472>572>622
                                                    +>366>471>571>621
                                           +>187>268>365>470>570>620
                                                    +>364>469>569>619
                                  +->53>107>151>230>315>417>520>598
                                                    +>314>416>519>597
                                           +>150>229>313>415>518>596
                                                    +>312>414>517>595
                                        +->74>111>161>249>334>438>540
                                                    +>248>333>437>539
                                           +>110>160>247>332>436>538
                                                    +>246>331>435>537
                                  +->52>154>235>320>424>526>602>630
                                                    +>423>525>601>629
                                           +>234>319>422>524>600>628
                                                    +>421>523>599>627
                                        +->73>109>159>245>330>434>536
                                                    +>244>329>433>535
                                           +>108>152>232>317>419>522
                                                    +>231>316>418>521
                                     +->39->51>153>233>318>420
                                           +->72>103>146>225
                           +->14->20->27->37
      +-->2-->4-->8
```

```
##############################################################
##                DECOMPOSITION DE L'ARBRE                  ##
##############################################################

**********      Noeud 0 **********

valeur: X1
liste goals:
                quicksort(X1,X2,•)
**********      Noeud 1 **********

valeur: X3.X5
liste goals:
                quicksort(X3,X4,X5,•)
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 2 **********

valeur: X6.X7.X8
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X6,X7,X8,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
```

```
**********     Noeud 3 **********

valeur: X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
               partition(X13,X12,X14,X15,•)
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 4 **********

valeur: X3.X4.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 5 **********

valeur: X18.X19.X20.X21
liste goals:
               [partition(X13,X12,X14,X15,•) = partition(X18,X19,X20,X21,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 6 **********

valeur: X22.X23.[(X24.X26.X27.X28)+(|X24.X26.X27.|X28)]
liste goals:
               inegal(X26,X23,•)
               partition(X27,X23,X28,X25,•)
               [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 7 **********

valeur: X29.X30.[(X32.X33.X34.X35)+(|X32.X33.X34.|X35)]
liste goals:
               inegal(X33,X30,•)
               partition(X34,X30,X31,X35,•)
               [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 8 **********

valeur: X1.X2
liste goals:
**********     Noeud 9 **********

valeur: X11.X12.X13.X14.X15.[(X16.X17)+(|X16.|X17)]
liste goals:
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 10       **********

valeur: X36.X37
liste goals:
               [inegal(X26,X23,•) = inegal(X36,X37,•),
X22.X23.[(X24.X26.X27.X28)+(|X24.X26.X27.|X28)] ]
               partition(X27,X23,X28,X25,•)
               [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X15,X16,X11,•)
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 11       **********

valeur: X11.X15.X16
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 12       **********
```

```
valeur: X23.X26.X27
liste goals:
                partition(X27,X23,X28,X25,•)
                [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 13        **********

valeur: X30.X33
liste goals:
                partition(X34,X30,X31,X35,•)
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 14        **********

valeur: X38.[(X39.X40)+(|X39.|X40)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X38,X39,X40,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 15        **********

valeur:
        [(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.
|X49)])]
liste goals:
                partition(X45,X44,X46,X47,•)
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 16        **********

valeur: X11.X14.X15.[(X16.X17.X23.X25.X27.X28)+(|X16.|X17.X23.X25.X27.X28)]
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 17        **********

valeur: X50.X51.X52.X53
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X50,X51,X52,X53,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 18        **********

valeur:
        [(X54.X55.[(X56.X58.X59.X60)+(|X56.X58.X59.|X60)])+(|X54.X55.[(X56.X58.|X59.X60)+(|X56.
[(X58.|X59.|X60)+(|X58)])])]
liste goals:
                inegal(X58,X55,•)
                partition(X59,X55,X60,X57,•)
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 19        **********

valeur:
        [(X61.X62.[(X64.X65.X66.X67)+(|X64.X65.X66.|X67)])+(|X61.X62.[(X64.X65.|X66.X67)+(|X64.
[(X65.|X66.|X67)+(|X65)])])]
```

```
liste goals:
                inegal(X65,X62,•)
                partition(X66,X62,X63,X67,•)
                [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 20        **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 21        **********

valeur: X68.X70.X71
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X68,X69,X70,X71,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 22        **********

valeur:
        [(X72.[(X74.X76.X77.X78)+(|X74.X76.X77.|X78)])+(|X72.[(X74.X76.|X77.X78)+(|X74.[(X76.|X
77.|X78)+(|X76)])])]
liste goals:
                inegal(X76,X73,•)
                partition(X77,X73,X78,X75,•)
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 23        **********

valeur:
        [(X79.[(X82.X83.X84.X85)+(|X82.X83.X84.|X85)])+(|X79.[(X82.X83.|X84.X85)+(|X82.[(X83.|X
84.|X85)+(|X83)])])]
liste goals:
                inegal(X83,X80,•)
                partition(X84,X80,X81,X85,•)
                [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 24        **********

valeur: X11.X12.X13.X14.X15.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 25        **********

valeur: X30.X31.X34.X35
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 26        **********

valeur: X86.X87
```

```
liste goals:
                [inegal(X58,X55,•) = inegal(X86,X87,•),
[(X54.X55.[(X56.X58.X59.X60)+(|X56.X58.X59.|X60)])+(|X54.X55.[(X56.X58.|X59.X60)+(|X56.[(X58.|
X59.|X60)+(|X58)])])] ]
                partition(X59,X55,X60,X57,•)
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 27        **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 28        **********

valeur: X11.X15
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 29        **********

valeur: [(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 30        **********

valeur: X88.X89
liste goals:
                [inegal(X76,X73,•) = inegal(X88,X89,•),
[(X72.[(X74.X76.X77.X78)+(|X74.X76.X77.|X78)])+(|X72.[(X74.X76.|X77.X78)+(|X74.[(X76.|X77.|X78
)+(|X76)])])] ]
                partition(X77,X73,X78,X75,•)
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 31        **********

valeur: X11.X15.X16
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 32        **********

valeur: X11.X15
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 33        **********

valeur: X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 34        **********

valeur:
       X11.X14.[(X15.[(X16.X17.X23.X25.X27.X28)+(|X16.|X17.X23.X25.X27.X28)])+(|X15.[(X16.X17.
X23.|X25.X27.X28)+(|X16.|X17.X23.|X25.X27.X28)])]
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
```

```
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 35      **********

valeur: X55.X58
liste goals:
              partition(X59,X55,X60,X57,•)
              [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
              [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X15,X16,X11,•)
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 36      **********

valeur: X62.X65
liste goals:
              partition(X66,X62,X63,X67,•)
              [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
              [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X15,X16,X11,•)
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 37      **********

valeur: X1
liste goals:
**********    Noeud 38      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 39      **********

valeur: X90.[(X91.X92)+(|X91.|X92)]
liste goals:
              [quicksort(X47,X48,X43,•) = quicksort(X90,X91,X92,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
              quicksort(X46,X42,X49,•)
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 40      **********

valeur: X93.X96.X97.[(X100.X101)+(|X100.|X101)]
liste goals:
              partition(X97,X96,X98,X99,•)
              quicksort(X99,X100,X95,•)
              quicksort(X98,X94,X101,•)
              [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
              quicksort(X46,X42,X49,•)
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 41      **********

valeur: X73.X76
liste goals:
              partition(X77,X73,X78,X75,•)
              [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
              quicksort(X47,X48,X43,•)
              quicksort(X46,X42,X49,•)
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 42      **********

valeur: X80.X83
liste goals:
              partition(X84,X80,X81,X85,•)
              [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
              quicksort(X47,X48,X43,•)
```

```
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 43        **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 44        **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 45        **********

valeur: X102.X103.X104
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X102,X103,X104,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 46        **********

valeur:
       [(X105.X107.X108.X109.[(X112.X113)+(|X112.|X113)])+(|X105.X107.[(X108.|X109.[(X112.X113
)+(|X112.|X113)])+(|X108.|X113)])]
liste goals:
                partition(X109,X108,X110,X111,•)
                quicksort(X111,X112,X107,•)
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 47        **********

valeur: X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 48        **********

valeur: X55.X57.X59.X60
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 49        **********

valeur: X62.X63.X66.X67
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 50        **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 51        **********

valeur: [(X43.X46.X47.X48)+(|X43.X46.X47.|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 52        **********

valeur: X96.X97.X98.X99
liste goals:
                quicksort(X99,X100,X95,•)
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 53        **********

valeur: X96.X97.X98
liste goals:
                quicksort(X99,X100,X95,•)
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 54        **********

valeur: X73.X75.X77.X78
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 55        **********

valeur: X80.X81.X84.X85
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 56        **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 57        **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 58        **********

valeur: X11.X14.X15.X16.X17
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 59        **********

valeur: X109.X110.X111
liste goals:
                quicksort(X111,X112,X107,•)
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 60        **********

valeur: X11.X14.X15.X16.X17
liste goals:
```

```
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 61        **********

valeur: X30.X35
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 62        **********

valeur: X55.X57
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 63        **********

valeur: X62.X67
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 64        **********

valeur: X73.X75
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 65        **********

valeur: X80.X85
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 66        **********

valeur: X30.X31
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 67        **********

valeur: X55.X60
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 68        **********
```

```
valeur: X62.X63
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 69        **********

valeur: X73.X78
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 70        **********

valeur: X80.X81
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 71        **********

valeur: X1
liste goals:
**********      Noeud 72        **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 73        **********

valeur: X46.[(X49.X95.X99.X100)+(|X49.[(X95.X99.X100)+(|X95.X99.|X100)])]
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 74        **********

valeur: [(X95.X99.X100)+(|X95.X99.|X100)]
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 75        **********

valeur: X44.X47.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 76        **********

valeur: X108.X111
liste goals:
                quicksort(X111,X112,X107,•)
```

```
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 77        **********

valeur: X44.X46.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 78        **********

valeur: X108.X110
liste goals:
                quicksort(X111,X112,X107,•)
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 79        **********

valeur: X1
liste goals:
**********      Noeud 80        **********

valeur: X1
liste goals:
**********      Noeud 81        **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 82        **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 83        **********

valeur: [(X107.X111.X112)+(|X107.X111.|X112)]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 84        **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 85        **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 86        **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 87        **********

valeur:
       X11.[(X14.[(X15.[(X16.X17.X23.X25.X27.X28)+(|X16.|X17.X23.X25.X27.X28)])+(|X15.[(X16.X1
7.X23.|X25.X27.X28)+(|X16.|X17.X23.|X25.X27.X28)])])+(|X14.[(X15.[(X16.X17.X23.X25.X27.|X28)+(
```

```
|X16.|X17.X23.X25.X27.|X28)])+(|X15.[(X16.X17.X23.|X25.X27.|X28)+(|X16.|X17.X23.|X25.X27.|X28)
])])]
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X22,X23,X24,X25,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 88        **********

valeur: X96.X97
liste goals:
                quicksort(X99,X100,X95,•)
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 89        **********

valeur: X30.X35
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 90        **********

valeur: X30
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 91        **********

valeur: X55
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X54,X55,X56,X57,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 92        **********

valeur: X62
liste goals:
                [partition(X34,X30,X31,X35,•) = partition(X61,X62,X63,X64,•), X30.X33 ]
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 93        **********

valeur: X73
liste goals:
                [partition(X45,X44,X46,X47,•) = partition(X72,X73,X74,X75,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 94        **********

valeur: X80
liste goals:
```

```
                [partition(X45,X44,X46,X47,•) = partition(X79,X80,X81,X82,•),
[(X41.X44.X45.[(X48.X49)+(|X48.|X49)])+(|X41.[(X44.|X45.[(X48.X49)+(|X48.|X49)])+(|X44.|X49)])
] ]
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 95        **********

valeur: X44.X47.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 96        **********

valeur: X44.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 97        **********

valeur: X108
liste goals:
                quicksort(X111,X112,X107,•)
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 98        **********

valeur: X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 99        **********

valeur: X30
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 100       **********

valeur: X30.X31
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 101       **********

valeur: X44.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 102       **********

valeur: X44.X46.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
```

```
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 103       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 104       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 105       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 106       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 107       **********

valeur: 1
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 108       **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 109       **********

valeur: 1
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 110       **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 111       **********

valeur: 1
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 112       **********

valeur: [(X43.X46.X47.X48)+(|X43.X46.X47.|X48.|X49)]
liste goals:
```

```
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 113       **********

valeur: [(X107.X111.X112)+(|X107.X111.|X112)]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 114       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 115       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 116       **********

valeur: [(X10.X14.X17.X107.X111.X112)+(|X10.X14.|X17.|X107.X111.|X112)]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 117       **********

valeur:
       [(X10.[(X14.[(X17.X107.X111.X112)+(|X17.|X107.X111.X112)])+(|X14.[(X17.X107.|X111.X112)
+(|X17.|X107.|X111.X112)])])+(|X10.[(X14.[(X17.X107.X111.|X112)+(|X17.|X107.X111.|X112)])+(|X1
4.[(X17.X107.|X111.|X112)+(|X17.|X107.|X111.|X112)])])]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 118       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 119       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 120       **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 121       **********

valeur:
       [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
```

```
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 122       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 123       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 124       **********

valeur: X11.X14.X15.X16.X17
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 125       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 126       **********

valeur: [(X95.X99.X100)+(|X95.X99.|X100)]
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 127       **********

valeur: 1
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 128       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 129       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 130       **********

valeur: X30
liste goals:
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 131       **********

valeur: X30
liste goals:
```

```
                [partition(X13,X12,X14,X15,•) = partition(X29,X30,X31,X32,•),
X9.X11.X12.X13.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 132       **********

valeur: X44.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 133       **********

valeur: X44.[(X48.X49)+(|X48.|X49)]
liste goals:
                quicksort(X47,X48,X43,•)
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 134       **********

valeur: [(X43.X46.X47.X48)+(|X43.X46.X47.|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 135       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 136       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 137       **********

valeur: [(X10.X14.X17.X107.X111.X112)+(|X10.X14.|X17.|X107.X111.|X112)]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 138       **********

valeur:
       [(X10.[(X14.[(X17.X107.X111.X112)+(|X17.|X107.X111.X112)])+(|X14.[(X17.X107.|X111.X112)
+(|X17.|X107.|X111.X112)])])+(|X10.[(X14.[(X17.X107.X111.|X112)+(|X17.|X107.X111.|X112)])+(|X1
4.[(X17.X107.|X111.|X112)+(|X17.|X107.|X111.|X112)])])]
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 139       **********

valeur: X11.X14.X15.X16.X17
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 140       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
```

```
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 141       **********

valeur: X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 142       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 143       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 144       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 145       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 146       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 147       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 148       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 149       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 150       **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 151       **********

valeur: 1
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
```

```
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 152       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 153       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 154       **********

valeur: X46.[(X49.X99.X100)+(|X49.X99)]
liste goals:
                quicksort(X98,X94,X101,•)
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 155       **********

valeur: X111
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 156       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 157       **********

valeur: X111
liste goals:
                quicksort(X110,X106,X113,•)
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 158       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 159       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 160       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 161       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 162       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 163       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 164       **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 165       **********

valeur:
        [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 166       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 167       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 168       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 169       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 170       **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
```

```
**********     Noeud 171      **********

valeur:
       [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
               [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 172      **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
               [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 173      **********

valeur:
       [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
               [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 174      **********

valeur: X1.X2
liste goals:
**********     Noeud 175      **********

valeur: X1
liste goals:
**********     Noeud 176      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 177      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 178      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 179      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 180      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 181      **********

valeur: X1.X2
liste goals:
**********     Noeud 182      **********
```

```
valeur: X1
liste goals:
**********     Noeud 183      **********

valeur: X10.X14.X17
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 184      **********

valeur: X14.X17
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 185      **********

valeur: X11.X14.X15.X16.X17
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 186      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 187      **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
               [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
               quicksort(X46,X42,X49,•)
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 188      **********

valeur: 1
liste goals:
               [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
               quicksort(X46,X42,X49,•)
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 189      **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
               [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
               quicksort(X46,X42,X49,•)
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 190      **********

valeur: 1
liste goals:
               [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
               quicksort(X46,X42,X49,•)
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 191      **********

valeur: X11.X14.X15.X16.X17
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 192      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
```

```
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 193       **********

valeur: X11.X14.X15.X16.X17
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 194       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 195       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 196       **********

valeur: X11.X12.X13.[(X16.X17)+(|X16.|X17)]
liste goals:
                quicksort(X15,X16,X11,•)
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 197       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 198       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 199       **********

valeur: [(X43.X47.X48)+(|X43.X47.|X48)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 200       **********

valeur: 1
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 201       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 202       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 203       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
```

```
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 204      **********

valeur: 1
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 205      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 206      **********

valeur: 1
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 207      **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
              [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 208      **********

valeur:
       [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
              [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 209      **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
              [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 210      **********

valeur:
       [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
              [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 211      **********

valeur: X10.X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 212      **********

valeur: X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 213      **********
```

```
valeur: X11.X14.X15.X16.X17
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 214      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 215      **********

valeur: X11.X14.X15.X16.X17
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 216      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
               quicksort(X14,X10,X17,•)
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 217      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 218      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 219      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 220      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 221      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 222      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 223      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 224      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 225       **********

valeur: X1
liste goals:
**********      Noeud 226       **********

valeur: X1
liste goals:
**********      Noeud 227       **********

valeur: X1
liste goals:
**********      Noeud 228       **********

valeur: X1
liste goals:
**********      Noeud 229       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 230       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 231       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 232       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 233       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 234       **********

valeur: [(X94.X98.X101)+(|X94.X98.|X101)]
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 235       **********

valeur: 1
liste goals:
                [quicksort(X47,X48,X43,•) = quicksort(X93,X94,X95,•),
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)] ]
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 236       **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 237       **********

valeur:
        [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 238       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 239       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 240       **********

valeur: [(X10.X14.X17.X106.X110.X113)+(|X10.X14.|X17.|X106.X110.|X113)]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 241       **********

valeur:
        [(X10.[(X14.[(X17.X106.X110.X113)+(|X17.X106.X110.|X113)])+(|X14.[(X17.X106.|X110.X113)
+(|X17.X106.|X110.|X113)])])+(|X10.[(X14.[(X17.|X106.X110.X113)+(|X17.|X106.X110.|X113)])+(|X1
4.[(X17.|X106.|X110.X113)+(|X17.|X106.|X110.|X113)])])]
liste goals:
                [quicksort(X15,X16,X11,•) = quicksort(X105,X106,X107,•),
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)] ]
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 242       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 243       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 244       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 245       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 246       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
```

```
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 247       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 248       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 249       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 250       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 251       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 252       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 253       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 254       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 255       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 256       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 257       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 258       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 259       **********
```

```
valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 260       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 261       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 262       **********

valeur: X114.[(X115.X116)+(|X115.|X116)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X114,X115,X116,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 263       **********

valeur:
       X117.[(X118.X119.X120.X121.[(X124.X125)+(|X124.|X125)])+(|X118.|X119.X120.X121.[(X124.X
125)+(|X124.|X125)])]
liste goals:
                partition(X121,X120,X122,X123,•)
                quicksort(X123,X124,X119,•)
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 264       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 265       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 266       **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 267       **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 268       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 269       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 270       **********
```

```
valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
              quicksort(X46,X42,X49,•)
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 |
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 271      **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
              quicksort(X46,X42,X49,•)
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 272      **********

valeur: X10.X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 273      **********

valeur: X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 274      **********

valeur: X10.X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 275      **********

valeur: X14.X17
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 276      **********

valeur: X11.X14.X15.X16.X17
liste goals:
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 277      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 278      **********

valeur: X11.X14.X15.X16.X17
liste goals:
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 279      **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
              quicksort(X14,X10,X17,•)
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 280      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 281      **********

valeur: 1
liste goals:
              [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 282      **********
```

```
valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 283       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 284       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 285       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 286       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 287       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 288       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 289       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 290       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 291       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 292       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 293       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 294       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
```

```
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 295       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 296       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 297       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 298       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 299       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 300       **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 301       **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 302       **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 303       **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 304       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 305       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 306       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 307       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 308       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 309       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 310       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 311       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 312       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 313       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 314       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 315       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 316       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 317       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 318       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 319       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 320       **********

valeur: X46.X47.[(X48)+(|X48.|X49)]
```

```
liste goals:
                quicksort(X46,X42,X49,•)
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 321       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 322       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 323       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 324       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 325       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 326       **********

valeur: [(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
liste goals:
                quicksort(X14,X10,X17,•)
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 327       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 328       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 329       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 330       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 331       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 332       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
```

```
**********     Noeud 333      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 334      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 335      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 336      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 337      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 338      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 339      **********

valeur; X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 340      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 341      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 342      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 343      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 344      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 345      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 346      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 347      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
```

```
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 348      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 349      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 350      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 351      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 352      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 353      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 354      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 355      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 356      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 357      **********

valeur: X120.X121.X122.X123
liste goals:
              quicksort(X123,X124,X119,•)
              quicksort(X122,X118,X125,•)
              [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 358      **********

valeur: X120.X121.X122
liste goals:
              quicksort(X123,X124,X119,•)
              quicksort(X122,X118,X125,•)
              [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
              [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
              [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********    Noeud 359      **********

valeur: X120.X121
liste goals:
```

```
                quicksort(X123,X124,X119,•)
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 360       **********

valeur: X1.X2
liste goals:
**********      Noeud 361       **********

valeur: X1
liste goals:
**********      Noeud 362       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 363       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 364       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 365       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 366       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 367       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 368       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 369       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 370       **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 371       **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 372       **********
```

```
valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 373       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 374       **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 375       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 376       **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 377       **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 378       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 379       **********

valeur: X10.X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 380       **********

valeur: X14.X17
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 381       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 382       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 383       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 384       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 385       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 386       **********
```

```
valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 387       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 388       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 389       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 390       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 391       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 392       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 393       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 394       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 395       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 396       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 397       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 398       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 399       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 400       **********

valeur: X1.X2
liste goals:
**********      Noeud 401       **********
```

```
valeur: X1
liste goals:
**********     Noeud 402      **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 403      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 404      **********

valeur: X3.X4.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 405      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 406      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 407      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 408      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 409      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 410      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 411      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 412      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 413      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 414      **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 415      **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 416      **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
```

```
**********     Noeud 417      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 418      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 419      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 420      **********

valeur: X1
liste goals:
**********     Noeud 421      **********

valeur: [(X42.X46.X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 422      **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 423      **********

valeur: [(X42.X46,X49)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 424      **********

valeur: [(X42.X46)+(|X42.X46.|X49)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X41,X42,X43,•), X11.X15.X16 ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 425      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 426      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 427      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 428      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 429      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 430      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 431       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 432       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 433       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 434       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 435       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 436       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 437       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 438       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 439       **********

valeur: X1
liste goals:
**********      Noeud 440       **********

valeur: X1
liste goals:
**********      Noeud 441       **********

valeur: X1
liste goals:
**********      Noeud 442       **********

valeur: X1
liste goals:
**********      Noeud 443       **********

valeur: X1
liste goals:
**********      Noeud 444       **********

valeur: X1
liste goals:
**********      Noeud 445       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 446       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 447       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 448       **********
```

```
valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 449       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 450       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 451       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 452       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 453       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 454       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 455       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 456       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 457       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 458       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 459       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 460       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 461       **********

valeur: [(X119.X123.X124)+(|X119.X123.|X124)]
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 462       **********

valeur: X123
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 463       **********

valeur: [(X119.X123.X124)+(|X119.X123.|X124)]
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 464       **********

valeur: 1
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 465       **********

valeur: [(X119.X123.X124)+(|X119.X123.|X124)]
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 466       **********

valeur: 1
liste goals:
                quicksort(X122,X118,X125,•)
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 467       **********

valeur: X1.X2
liste goals:
**********      Noeud 468       **********

valeur: X1
liste goals:
**********      Noeud 469       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 470       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 471       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 472       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 473       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 474       **********

valeur: 1
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
```

```
**********     Noeud 475      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 476      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 477      **********

valeur: X1.X2
liste goals:
**********     Noeud 478      **********

valeur: X1
liste goals:
**********     Noeud 479      **********

valeur: X1.X2
liste goals:
**********     Noeud 480      **********

valeur: X1
liste goals:
**********     Noeud 481      **********

valeur: X3.X4.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 482      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 483      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 484      **********

valeur: X3.X4.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 485      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 486      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 487      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 488      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 489      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 490      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 491      **********

valeur: X3.X5
```

```
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 492       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 493       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 494       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 495       **********

valeur: X1
liste goals:
**********      Noeud 496       **********

valeur: X1
liste goals:
**********      Noeud 497       **********

valeur: X1
liste goals:
**********      Noeud 498       **********

valeur: X1
liste goals:
**********      Noeud 499       **********

valeur: X1
liste goals:
**********      Noeud 500       **********

valeur: X1
liste goals:
**********      Noeud 501       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 502       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 503       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 504       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 505       **********

valeur: X1.X2
liste goals:
**********      Noeud 506       **********

valeur: X1
liste goals:
**********      Noeud 507       **********

valeur: X1.X2
liste goals:
**********      Noeud 508       **********

valeur: X1
liste goals:
**********      Noeud 509       **********

valeur: X1
liste goals:
```

```
**********     Noeud 510      **********

valeur: X1
liste goals:
**********     Noeud 511      **********

valeur: X1
liste goals:
**********     Noeud 512      **********

valeur: X1
liste goals:
**********     Noeud 513      **********

valeur: X1
liste goals:
**********     Noeud 514      **********

valeur: X1
liste goals:
**********     Noeud 515      **********

valeur: X1
liste goals:
**********     Noeud 516      **********

valeur: X1
liste goals:
**********     Noeud 517      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 518      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 519      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 520      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 521      **********

valeur: X1
liste goals:
**********     Noeud 522      **********

valeur: X1
liste goals:
**********     Noeud 523      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 524      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 525      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 526      **********

valeur: 1
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 527      **********
```

```
valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 528       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 529       **********

valeur: X1
liste goals:
**********      Noeud 530       **********

valeur: X1
liste goals:
**********      Noeud 531       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 532       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 533       **********

valeur: X1
liste goals:
**********      Noeud 534       **********

valeur: X1
liste goals:
**********      Noeud 535       **********

valeur: X1
liste goals:
**********      Noeud 536       **********

valeur: X1
liste goals:
**********      Noeud 537       **********

valeur: X1
liste goals:
**********      Noeud 538       **********

valeur: X1
liste goals:
**********      Noeud 539       **********

valeur: X1
liste goals:
**********      Noeud 540       **********

valeur: X1
liste goals:
**********      Noeud 541       **********

valeur: X1
liste goals:
**********      Noeud 542       **********

valeur: X1
liste goals:
**********      Noeud 543       **********

valeur: X1
liste goals:
**********      Noeud 544       **********

valeur: X1
liste goals:
**********      Noeud 545       **********

valeur: X1
liste goals:
**********      Noeud 546       **********
```

```
valeur: X1
liste goals:
**********     Noeud 547      **********

valeur: X1
liste goals:
**********     Noeud 548      **********

valeur: X1
liste goals:
**********     Noeud 549      **********

valeur: X1
liste goals:
**********     Noeud 550      **********

valeur: X1
liste goals:
**********     Noeud 551      **********

valeur: X1
liste goals:
**********     Noeud 552      **********

valeur: X1
liste goals:
**********     Noeud 553      **********

valeur: X1
liste goals:
**********     Noeud 554      **********

valeur: X1
liste goals:
**********     Noeud 555      **********

valeur: X1
liste goals:
**********     Noeud 556      **********

valeur: X1
liste goals:
**********     Noeud 557      **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 558      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 559      **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 560      **********

valeur: 1
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 561      **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
               [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 562       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 563       **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 564       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 565       **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 566       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 567       **********

valeur: [(X118.X122.X125)+(|X118.X122.|X125)]
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 568       **********

valeur: 1
liste goals:
                [quicksort(X14,X10,X17,•) = quicksort(X117,X118,X119,•),
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)] ]
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 569       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 570       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 571       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 572       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 573       **********

valeur: X3.X5
liste goals:
```

```
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 574       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 575       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 576       **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 577       **********

valeur: X1.X2
liste goals:
**********      Noeud 578       **********

valeur: X1
liste goals:
**********      Noeud 579       **********

valeur: X1
liste goals:
**********      Noeud 580       **********

valeur: X1.X2
liste goals:
**********      Noeud 581       **********

valeur: X1
liste goals:
**********      Noeud 582       **********

valeur: X1
liste goals:
**********      Noeud 583       **********

valeur: X1
liste goals:
**********      Noeud 584       **********

valeur: X1
liste goals:
**********      Noeud 585       **********

valeur: X1
liste goals:
**********      Noeud 586       **********

valeur: X1
liste goals:
**********      Noeud 587       **********

valeur: X1
liste goals:
**********      Noeud 588       **********

valeur: X1
liste goals:
**********      Noeud 589       **********

valeur: X1
liste goals:
**********      Noeud 590       **********

valeur: X1
liste goals:
**********      Noeud 591       **********

valeur: X1
liste goals:
**********      Noeud 592       **********

valeur: X1
liste goals:
**********      Noeud 593       **********
```

```
valeur: X1
liste goals:
**********     Noeud 594      **********

valeur: X1
liste goals:
**********     Noeud 595      **********

valeur: X1
liste goals:
**********     Noeud 596      **********

valeur: X1
liste goals:
**********     Noeud 597      **********

valeur: X1
liste goals:
**********     Noeud 598      **********

valeur: X1
liste goals:
**********     Noeud 599      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 600      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 601      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 602      **********

valeur: X3.X5
liste goals:
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 603      **********

valeur: X1
liste goals:
**********     Noeud 604      **********

valeur: X1
liste goals:
**********     Noeud 605      **********

valeur: X1
liste goals:
**********     Noeud 606      **********

valeur: X1
liste goals:
**********     Noeud 607      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 608      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 609      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
               [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
               [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 610      **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
```

```
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 611       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 612       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 613       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 614       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 615       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 616       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 617       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 618       **********

valeur: [(X10.X14.X17)+(|X10.X14.|X17)]
liste goals:
                [quicksort(X3,X4,X5,•) = quicksort(X9,X10,X11,•), X3.X5 ]
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********      Noeud 619       **********

valeur: X1
liste goals:
**********      Noeud 620       **********

valeur: X1
liste goals:
**********      Noeud 621       **********

valeur: X1
liste goals:
**********      Noeud 622       **********

valeur: X1
liste goals:
**********      Noeud 623       **********

valeur: X1
liste goals:
**********      Noeud 624       **********

valeur: X1
liste goals:
**********      Noeud 625       **********

valeur: X1
liste goals:
**********      Noeud 626       **********

valeur: X1
```

```
liste goals:
**********     Noeud 627      **********

valeur: X1
liste goals:
**********     Noeud 628      **********

valeur: X1
liste goals:
**********     Noeud 629      **********

valeur: X1
liste goals:
**********     Noeud 630      **********

valeur: X1
liste goals:
**********     Noeud 631      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 632      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 633      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 634      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 635      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 636      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 637      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 638      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 639      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 640      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 641      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 642      **********

valeur: X3.X5
liste goals:
                [quicksort(X1,X2,•) = quicksort(X3,X4,•), X1 ]
**********     Noeud 643      **********

valeur: X1
liste goals:
```

```
**********     Noeud 644      **********

valeur: X1
liste goals:
**********     Noeud 645      **********

valeur: X1
liste goals:
**********     Noeud 646      **********

valeur: X1
liste goals:
**********     Noeud 647      **********

valeur: X1
liste goals:
**********     Noeud 648      **********

valeur: X1
liste goals:
**********     Noeud 649      **********

valeur: X1
liste goals:
**********     Noeud 650      **********

valeur: X1
liste goals:
**********     Noeud 651      **********

valeur: X1
liste goals:
**********     Noeud 652      **********

valeur: X1
liste goals:
**********     Noeud 653      **********

valeur: X1
liste goals:
**********     Noeud 654      **********

valeur: X1
liste goals:

==============         Ligne 1       ==============
quicksort(X1,X2,•)
X1
----- 2 solutions
X1.X2
X1
==============         Ligne 2       ==============
quicksort(X3,X4,X5,•)
X3.X5
----- 2 solutions
X3.X4.X5
X3.X5
==============         Ligne 3       ==============
partition(X13,X12,X14,X15,•)
X12.X13
----- 3 solutions
X11.X12.X13.X14.X15.[(X16.X17)+(|X16.|X17)]
X11.X12.X13.X14.[(X16.X17)+(|X16.|X17)]
X11.X12.X13.[(X16.X17)+(|X16.|X17)]
==============         Ligne 4       ==============
inegal(X26,X23,•)
X23.X26
----- 1 solutions
X23.X26.X27
==============         Ligne 5       ==============
quicksort(X14,X10,X17,•)
1
----- 2 solutions
[(X10.X14.X17)+(|X10.X14.|X17)]
1
==============         Ligne 6       ==============
partition(X34,X30,X31,X35,•)
X30
----- 4 solutions
X30.X31.X34.X35
X30.X35
```

```
X30.X31
X30
===============         Ligne 7       ==============
partition(X45,X44,X46,X47,•)
1
----- 4 solutions
[(X44.X45.X46.X47.[(X48.X49)+(|X48.|X49)])+(|X44.X45.X46.X47.|X49)]
X44.X47.[(X48.X49)+(|X48.|X49)]
X44.X46.[(X48.X49)+(|X48.|X49)]
X44.[(X48.X49)+(|X48.|X49)]
===============         Ligne 8       ==============
inegal(X58,X55,•)
X55
----- 1 solutions
X55.X58
===============         Ligne 9       ==============
inegal(X76,X73,•)
1
----- 1 solutions
X73.X76
===============         Ligne 10      ==============
quicksort(X47,X48,X43,•)
X47
----- 2 solutions
[(X43.X46.X47.X48)+(|X43.X46.X47.|X48.|X49)]
X46.X47.[(X48)+(|X48.|X49)]
===============         Ligne 11      ==============
quicksort(X15,X16,X11,•)
X11
----- 2 solutions
X11.X14.X15.X16.X17
[(X10.X11.X14.X16.X17)+(|X10.X11.X14.|X16.|X17)]
===============         Ligne 12      ==============
quicksort(X14,X10,X17,•)
[(X10.X14.X17)+(|X10.X14.|X17)]
----- 1 solutions
[(X10.X14.X17)+(|X10.X14.|X17)]
```